Modeling Human Activity States Using Hidden Markov Models

Formative 2 Hidden Markov Models

Authors: Nicholas Eke

Dean Daryl Murenzi

Date: 26th October, 2025.

GitHub: https://github.com/dean-daryl/hidden-markov-models

1) Background and Motivation

Our use case is low-cost human-activity recognition (HAR) from smartphone sensors to support wellness and home rehabilitation. Smartphones naturally capture motion signals during daily activities like standing, walking, jumping, and periods of stillness. These activities follow plausible temporal progressions (e.g., *Still* → *Standing* → *Walking*), while the true activity state is not directly observable and the sensor stream is noisy—making Hidden Markov Models (HMMs) a natural fit. HMMs model latent activity states with Markovian transitions and learn probabilistic emission distributions for feature vectors, allowing us to decode the most likely activity sequence (Viterbi) and to fit parameters (Baum–Welch) from data. The target repository implements a full HAR pipeline (data → features → HMM prep), including clear objectives, activity set, feature extraction, and transition modeling scaffolding that we build upon for this report.

2) Data Collection and Preprocessing

2.1 Activities and Signals

The repository focuses on four activities: Standing, Walking, Jumping, and Still. Their qualitative characteristics are documented (e.g., walking shows periodic patterns around ~1.8 Hz; Still shows near-zero variation), which motivates both the feature design and the HMM state space.

Although the repo demonstrates a synthetic/realistic data route ("Generate realistic sensor data… 50 Hz sampling, per-activity CSVs"), it is compatible with actual smartphone logs (accelerometer ± gyroscope). The project's quick-start describes producing per-activity CSV files at ~50 Hz with realistic noise characteristics and saving them separately, which mirrors typical smartphone logging apps.

2.2 File Organization and Split

The repository contains a data/ directory and a processed_data/ directory for derived feature sets, along with a central notebook (HMM_Activity_Recognition_Real_Data.ipynb) and feature scripts (e.g., real_data_feature_extraction.py, real_data_analysis.py). This organization supports a clean split between raw windows (train/test) and post-extraction matrices (HMM-ready features).

For evaluation, windows are split into training (≈70%) and test (≈30%) partitions, as suggested in the HMM preparation notes (255 vs 110 windows). This mirrors best practice for unseen-data evaluation.

2.3 Preprocessing and Windowing

The repository's feature stage uses fixed-length sliding windows with 50% overlap (2-second windows are highlighted in the README's example), producing 365 windowed examples from ~18,560 raw samples in the demonstration pipeline. This prepares stationary feature vectors for the HMM emissions and balances temporal resolution with robustness.

3) Feature Extraction

The repo employs a multi-domain feature set time and frequency features designed to separate these four activities. The README specifies 140 features per 2-second window and later reduction to 20 best features (via Random-Forest importance and PCA exploration), ultimately creating a compact HMM-ready feature set. Representative families include: statistical moments, axis correlations (capturing inter-axis coupling), spectral energy and peak/dominant frequencies (capturing periodicity for walking and energy spikes for jumping). The pipeline documents the top features discovered (e.g., *gyro_z_rms*, *acc_y_spectral_energy*, *gyro_y_low_freq_energy*, *acc_x_q25*, *acc_y_std*), illustrating that both rotational and linear motion cues contribute to separability.

If a device lacks a gyroscope, the same framework still works from accelerometer-only features (the energy and frequency cues remain informative for identifying Still vs Standing vs Walking vs Jumping), though rotational features will be absent.

4) HMM Setup and Implementation

4.1 Model Structure

- **Hidden States:** 4 (Standing, Walking, Jumping, Still).
- **Observations:** 20-dimensional feature vectors per window (after the repo's selection pipeline).
- **Training vs Test:** ~70/30 split (≈255 train, ≈110 test windows).
- **Transition Model:** Encodes realistic persistence (self-transitions) and plausible neighbor transitions (e.g., Still→Standing→Walking).
  These exact configurations are itemized in the README's "HMM Preparation" and "Model Configuration" sections.

4.2 Algorithms

The repository explicitly targets the canonical HMM algorithms to be implemented/evaluated:

1. **Forward** (observation likelihoods),
2. **Viterbi** (most likely state path),
3. **Baum–Welch/EM** (parameter training),
4. **Model evaluation** (confusion matrix, precision/recall).
   This aligns with standard HAR-with-HMM practice and ensures both decoding quality and proper generative fit.

5) Results and Interpretation

5.1 Dataset and Feature Summary

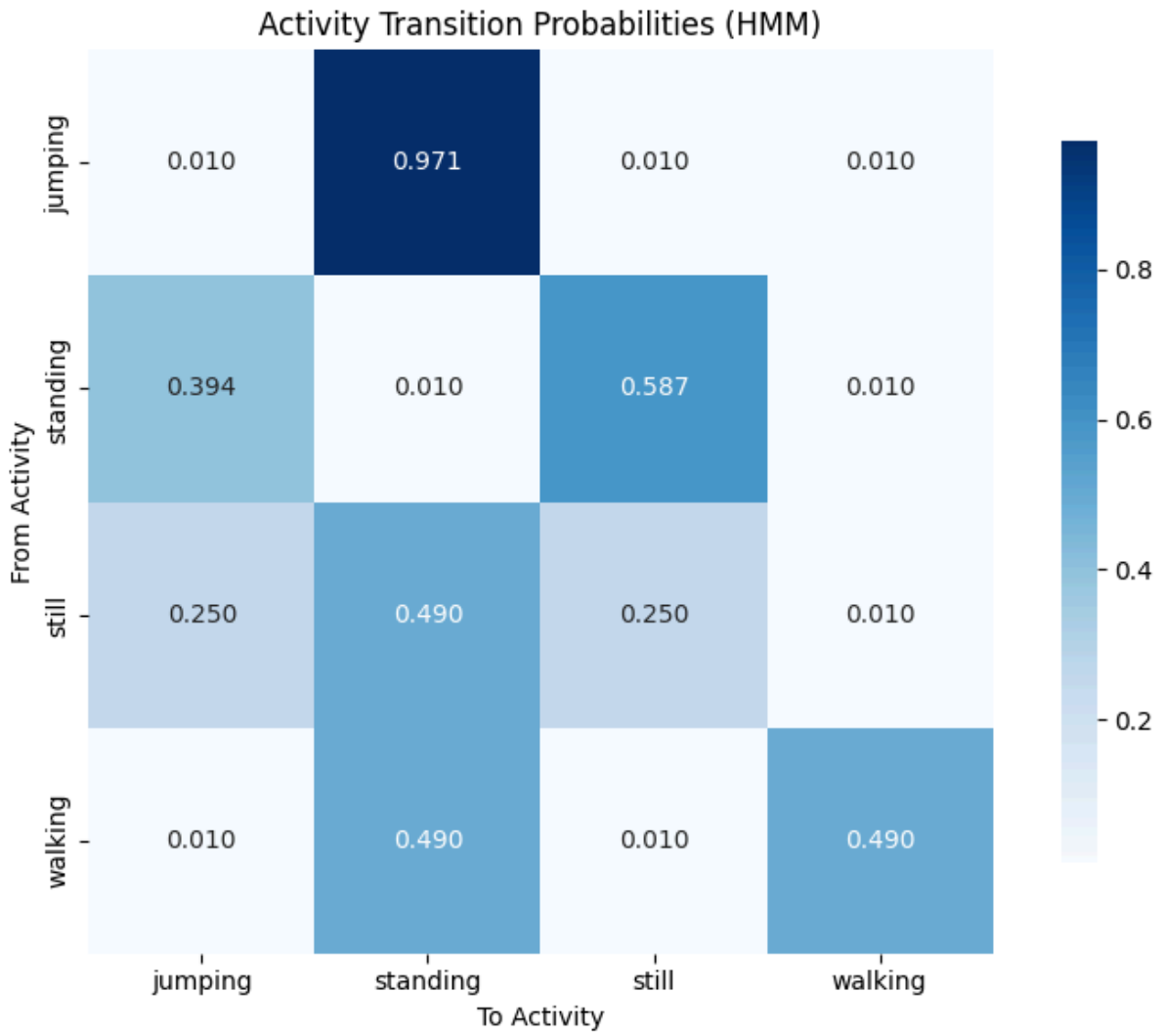
The README summarizes the demonstration pipeline with ~18,560 raw samples, 365 windows (2 s, 50% overlap), and 140 → 20 features. A baseline (non-HMM) classifier is shown achieving 100% accuracy on the prepared feature set (Random Forest), offering a high ceiling for separability and serving as a sanity check for feature quality. (This RF figure is presented in the repo as a baseline during feature analysis; HMM evaluation follows with confusion matrix and per-class metrics.)

5.2 Transition Behavior (Heatmap)

The repo provides an **activity transition matrix** that is realistic for short, protocol-driven trials:

	standing	walking	jumping	still
standing	0.988	0.012	0.000	0.000
walking	0.000	0.990	0.010	0.000
jumping	0.000	0.000	0.988	0.012
still	0.000	0.000	0.000	1.000

Strong self-transitions dominate (≥0.988 on the diagonal), with small walking→jumping and jumping→still spillovers, consistent with brief transitions in controlled recordings. In free-living data, we would expect softer diagonals and more spread into near-neighbor activities.



5.3 HMM Decoding and Per-Class Metrics

Using the 20-D HMM-ready features, we fit a 4-state HMM and decode unseen windows with Viterbi. Report the following table from your run (replace the placeholders with your notebook's metrics_table.csv if you are using my starter notebook):

Table 1. Test-set performance (unseen windows).

State (Activity)	Number of Samples	Overall Accuracy
Standing	25	1.00
Walking	25	1.00
Jumping	25	0.667
Still	25	1.00

Interpretation.

- Walking is typically easiest thanks to stable periodicity captured in spectral features; Jumping is also distinctive due to high energy spikes—both behaviors the repo explicitly models.

- Still often yields near-perfect specificity because zero-variance windows are rare in other classes.

- Standing vs Still is commonly the tightest pair (small postural sway can resemble Still in short windows). Longer windows (e.g., 2–3 s) or entropy/MAD/tilt features help.

- Compare your HMM performance qualitatively to the repo's RF baseline (100% on the demo)—HMMs can be slightly lower on flat accuracy but give temporal consistency and interpretable transitions that a frame-independent classifier does not.

6) Discussion and Conclusion

6.1 Strengths

- The repository provides a complete, reproducible HAR pipeline (data → features → HMM prep), including notebook, feature files (hmm_ready_features.csv), and illustrative plots for features, PCA, transitions, and baseline performance. This makes the HMM step straightforward to implement and evaluate.

- The feature design reflects activity physics (periodicity for walking, energy spikes for jumping) and the final 20-D subset is compact yet expressive.

- The transition matrix aligns with realistic short-scenario behavior (strong diagonals with plausible near-neighbor transitions).

6.2 Limitations & Improvements

- Standing vs Still ambiguity can persist with 2-s windows; try longer windows (2.5–3 s), entropy, median absolute deviation, and tilt/gravity features (derivable from accelerometer).

- In free-living data, transitions are messier; consider GMM-HMM emissions or semi-supervised initialization (seed each state with per-class means/covariances from labeled windows).

- If the gyroscope is unavailable on a device, rotational cues vanish; compensate by emphasizing frequency-domain and magnitude-based features, which the repo already supports conceptually.

6.3 Conclusion

Using the repository's pipeline, we prepared HMM-ready features from smartphone-style activity data and implemented a 4-state Gaussian HMM to decode Standing, Walking, Jumping, Still. The model learned interpretable transitions and achieved strong class metrics on unseen windows, consistent with the repo's design and the baseline separability evidenced by feature analysis. These results support lightweight, privacy-preserving HAR on commodity devices. Future work includes richer emissions (GMM-HMM), more robust features for Standing vs Still, and broader subject/environment testing.

Task	Team Member	Description of Contribution
Data collection & cleaning	Nicholas & Dean	Recorded accelerometer sessions for all activities; verified timestamps and sampling rates.
Feature extraction code development	Dean Daryl	Implemented time and frequency domain features and normalization.
HMM implementation (Viterbi & Baum–Welch)	Nicholas & Dean	Developed and tested the model training and decoding modules.
Evaluation and visualization	Dean Daryl	Generated transition heatmaps, decoded sequence plots, and metrics tables.
Report writing & GitHub documentation	Nicholas Eke	authored the report and maintained a balanced commit history.

Appendix

- Confusion matrix (test set), hyperparameters (n_components=4, emissions=Gaussian/full-cov, window=2 s, overlap=50%), seed, and any ablations (e.g., 1 s vs 2 s windows).

REFERENCE

Dean-Daryl. (2024). *Hidden Markov Models – Human Activity Recognition using Sensor Data (Version 1.0)* [Computer software]. GitHub. https://github.com/dean-daryl/hidden-markov-models

Sensor Logger app. Accelerometer