Human Activity Recognition (HAR)

Github repo: https://github.com/lmurayire12/Hidden-Markov-Models.git

Background and motivation

Our group built a context-aware safety assistant for elder-care environments. The assistant monitors resident activity using a body-worn sensor (accelerometer + gyroscope) to detect unsafe conditions (falls, prolonged inactivity) and to log behavior patterns. Human Activity Recognition (HAR) is needed to convert raw sensor streams into reliable activity labels so the assistant can trigger appropriate interventions (alerts for suspected falls, reminders after excessive inactivity, or activity logs for clinicians). Accurate HAR reduces false alarms and improves situational awareness for caregivers.

Summary

We built and evaluated an HMM-based pipeline that maps short windows of accelerometer+gyroscope data to four activity labels: `walking`, `standing`, `still`, `jumping`.

- Data: processed per-activity CSVs in `data_processed/`; windowed features saved to `data_processed/all_features.csv` (370 windows total).
- Key per-window results:
  - Baseline unsupervised HMM (majority-vote mapping): overall ≈ 77.03% (single-split).
  - Supervised-initialized HMM (seeded with class statistics): overall ≈ 97.30% (single-split) and mean ≈ 99.73% across stratified 5-fold CV.

This brief report contains the method, core results, artifacts, and concise next steps.

Data & preprocessing

Files used: the four processed CSVs under `data_processed/`. We extract non-overlapping windows of 50 samples (≈2 s at ~50 Hz) and compute per-window features: per-axis mean/std, SMA, pairwise axis correlations, and simple frequency-domain descriptors (dominant index, spectral energy, top FFT magnitudes). The final feature file is `data_processed/all_features.csv` (370 windows: jumping 85, standing 84, still 106, walking 95). We Z-score features before modeling.

Modeling approach

We used Gaussian HMMs (`hmmlearn`) with 4 components and diagonal covariances. Two workflows were evaluated:

1) Baseline (unsupervised): Baum–Welch on training features; map latent states→labels by majority vote on training windows.

2) Supervised initialization: seed each component's mean/diag-cov from class statistics and seed start/transition probabilities from label frequencies, then run EM from these initials.

Majority-vote mapping is convenient but can be unstable in extreme hold-out tests where a class is absent from training.

Evaluations performed (core numbers)

- Single stratified 80/20 split:
  - Baseline unsupervised HMM (test): ≈ 77.03%.
  - Supervised-initialized HMM (test): ≈ 97.30%.
- Stratified 5-fold CV (per-fold results saved to `results/stratified_kfold.csv`): mean baseline ≈ 77.30%, mean supervised ≈ 99.73% (fold accuracies are available in the CSV).
- Leave-one-activity-out (LOAO): per-window accuracy was 0.0 for held-out classes under our mapping protocol. LOAO is a strict unseen-class test; majority-vote mapping and per-window accuracy are not meaningful here without label-aware strategies.

Per-class (single-split supervised)

Saved in `results/per_class_metrics_supervised.csv`. Highlights (test support in parentheses):

- jumping: precision=1.00, recall=1.00 (17)
- standing: precision≈0.94, recall≈0.94 (17)
- still: precision=1.00, recall≈0.95 (21)
- walking: precision≈0.95, recall=1.00 (19)

Supervised initialization closes the gap for classes that the unsupervised mapping sometimes missed.

Limitations & notes

- Small dataset (370 windows): per-window metrics are informative but not definitive for deployment. We should prioritize subject- or session-level splits for stronger claims.
- LOAO measured here is an extreme unseen-class test; per-window accuracy is not a fair metric without label-aware strategies.
- We observed occasional EM convergence warnings; better initialization, regularization, or increased iterations help.
- Covariance shapes from `hmmlearn` can vary by version; use a small wrapper to save/load consistently.

Recommended next steps (to get useful sequence-level generalization estimates)

1. Instead of leave-one-activity-out, we should run stratified k-fold or leave-one-recording-out (if we have multiple recordings/subjects) so each fold contains windows of all classes in train and test. This better estimates typical generalization performance.

2. If we do want to evaluate unseen activities, change the metric to a semantic/cluster-level similarity (e.g., measure whether the held-out activity maps to any learned latent component consistently) rather than per-window label accuracy.

3. For LOAO with label prediction, consider using label-informed initialization and freeze mapping (or evaluate by clustering distances) so the held-out class can be assigned to the closest existing component rather than expecting exact label match.

Stratified 5-fold comparison: baseline vs supervised-initialized

We ran a stratified 5-fold evaluation (preserving class proportions in each fold). The per-fold accuracies were saved to `results/stratified_kfold.csv` and a detailed JSON to `results/stratified_kfold_summary.json`.

Per-fold results (from `results/stratified_kfold.csv`):

fold	n_test_windows	baseline_acc	supervised_acc
1	74	0.783784	1.000000
2	74	0.770270	1.000000
3	74	0.770270	1.000000
4	74	0.770270	0.986486
5	74	0.770270	1.000000

Overall (mean) accuracy across folds:

- Baseline (unsupervised HMM): 0.772973 (≈77.30%)
- Supervised-initialized HMM: 0.997297 (≈99.73%)

Interpretation : The stratified k-fold evaluation keeps all labels present in each fold; this produces stable state-to-label mappings and is a better indicator of per-window predictive performance than the leave-one-activity-out test for the current setup. - The supervised-initialized HMM shows a marked improvement in per-window accuracy in this evaluation, consistent with earlier single-split supervised results in this repository. Expect near-perfect per-window classification when class-specific emission means and variances are good initial estimates.