Human Activity Recognition with HMMs

William Jow
2nd year undergraduate
williamjow@berkeley.edu

David Lin
3rd year undergraduate
lin.david@berkeley.edu

Owen Jow
4th year undergraduate
owenjow@berkeley.edu

Abstract

In this project, we explore applications of hidden Markov models (HMMs) in the context of human activity recognition. We frame the recognition problem as a multinomial classification instance, where our model is given a video as input and must associate it with one of a predetermined set of activities. In building a model, we first extract per-sequence feature matrices from a training corpus of videos and subsequently run the EM algorithm on per-class HMMs in order to fit their parameters to each class's features. Later, we extend MLE-based evaluation using normalization and ensemble methods. Our algorithm produces models with 50-80% classification accuracy on a test set, vastly outperforming the baseline 17% given by uniform random sampling.

1 Introduction

Activity recognition is an active area of research in computer vision, and has applications in a wide range of domains (including healthcare, HCI, and real-time surveillance carried out by oppressive totalitarian regimes). In the modern era, typical cutting-edge methods (e.g. [1], [2], [3]) ride the deep learning wave and develop fancy statistical ML formulations that involve buzzwords such as "convolutional neural networks" and "LSTMs." In this report, however, we take a step backward and argue that *Markov chains are all you need* [1], training a hidden Markov model (HMM)-based classifier on a dataset of a few hundred videos in order to achieve surprisingly decent results.

More specifically, we consider (human) activity recognition as a multinomial classification problem and build a model based on multiple HMMs, one for each activity, where the job of the aggregate model is to take a video of a human doing something and decide whether the human is (a) walking, (b) running, or so on. Each HMM is trained to represent the sequential progression of a single activity.

To engineer each HMM, we first extract centroid, edge, and optical flow features from all training videos corresponding to that HMM's activity, and then run the expectation-maximization (EM) algorithm to determine the MLE of the HMM's parameters given the set of observed feature vectors. After training each HMM in this fashion, we can evaluate the overall model on a video by having each HMM score the video's feature representation and then choosing the activity whose HMM produces the highest score. We also find slight improvement through normalization and ensemble methods, in which we (a) attempt to enforce that the output distributions of each HMM resemble each other in first and second central moments and (b) train multiple HMMs for each activity and average their outputs.

We evaluate our algorithm on the KTH action dataset [4], which contains several hundred videos of six activity classes over homogeneous backgrounds. In contrast to uniform random sampling, which gives an expected 17% classification accuracy, our method classifies each activity with 50-80% accuracy on a separate test set. Meanwhile, despite some variability in performance, ensemble models are shown to slightly outperform vanilla models in classifying most activities.

---

[1]to attain 50% accuracy

2 Background

2.1 HMMs

Our method uses hidden Markov models in order to predict the correct activity. In our context, the hidden Markov model (HMM) is a sequence classifier that assigns a label to each item in a sequence. An HMM involves an underlying Markov chain, which is a sequence of variables in which the probability of an event is dependent only on the prior state. Given a sequence of observations, the HMM computes a probability distribution over the possible label sequences and selects the best one.

An HMM relies on three components: a transition probability distribution, an observation likelihood distribution, and an initial state distribution. The transition distribution describes the distribution for the next state given the current state. The observation distribution describes the distribution of the output given the current state. Finally, the initial state distribution describes the starting distribution over the states. Together, these properties allow the HMM to compute the probability for any sequence of events.

An HMM also has two core algorithms to help determine the likelihood of an observation given an HMM and to learn the HMM parameters.

2.2 The Forward Algorithm

The forward algorithm is used to compute the likelihood of a certain observation sequence. The algorithm uses dynamic programming to compute the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence. To compute the forward probability, the previous forward path probability, the transition probability, and the state observation likelihood are multiplied together in each step. The pseudocode of this algorithm is as follows:

> If $\alpha_t(j) = \texttt{forward}[j, t]$ denotes the probability of being in state j after observing the first t observations, $a_{i,j}$ denotes the transition probability from state i to state j, $b_j(o_t)$ denotes the probability of observing o_t given that the current state is j, and s_F is the final state,

---
Algorithm 1 Forward Algorithm

---
procedure FORWARD(observations length T, state graph length N)
    initialize $\texttt{forward}[N, T]$
    for each state s from 1 to N **do**
        $\texttt{forward}[s, 1] \leftarrow a_{0,s} \cdot b_s(o_1)$
    for each time step t from 2 to T **do**
        for each state s from 1 to N **do**
            $\texttt{forward}[s, t] \leftarrow \sum_{s'=1}^{N} \texttt{forward}[s', t-1] \cdot a_{s',s} \cdot b_s(o_t)$
    return $\sum_{s=1}^{N} \texttt{forward}[s, T] \cdot a_{s,s_F}$

> In our implementation, we take the log of each probability and thus perform addition where there is multiplication above.

2.3 The Forward-Backward Algorithm

The forward-backward (or Baum-Welch) algorithm, a special case of the expectation-maximization (EM) algorithm, is used to learn the parameters of an HMM. The algorithm allows us to train the transition and emission probabilities of the HMM. It iteratively improves on the probability estimates that it learns. Given a sequence of observations and emissions, the algorithm computes the posterior marginals of all hidden state variables. It relies on calculating the forward and backward probabilities in order to compute the transition probability and observation probability. The pseudocode of this algorithm is as follows:

If $\xi_t(i, j)$ denotes the probability of being in state i at time t and state j at time $t + 1$, $\gamma_t(j)$ denotes the probability of being in state j at time t, and backward probability $\beta_t(i)$ denotes the probability of observing the observations from time $t + 1$ onward given that the current state is i,

---

Algorithm 2 Forward-Backward Algorithm

---

procedure FORWARD-BACKWARD(observations length T, observation set V)
    initialize transition matrix A, observation distribution B
    repeat

        for all t, j **do** $\gamma_t(j) = \dfrac{\alpha_t(j)\beta_t(j)}{\alpha_T(s_F)}$

        for all t, i, j **do** $\xi_t(i, j) = \dfrac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(s_F)}$

$$A_{ij} \leftarrow \left(\sum_{t=1}^{T-1} \xi_t(i, j) \right) \Big/ \left(\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i, k) \right)$$

$$B_j(v_k) \leftarrow \left(\sum_{\substack{t=1 \\ \text{s.t. } o_T = v_k}}^{T} \gamma_t(j) \right) \Big/ \left(\sum_{t=1}^{T} \gamma_t(j) \right)$$

    until convergence

---

3 Methods

3.1 Overview

The activity recognition pipeline involves three main components: feature extraction (*data processing*), model construction (*training*), and video classification (*evaluation*).

1. *Feature extraction.* Subtract the background, determine regions of interest, and then extract shape and optical flow features.
2. *Model construction.* Pass these features to a k-means clustering algorithm, which provides an alignment for each time-indexed feature with one of k states. Use the aligned sequences to train HMMs with the Baum-Welch algorithm, one HMM for each activity.
3. *Evaluation.* At test time: perform MLE with our pre-trained HMMs.

In the following sections, we detail the methodology behind each of these components.

3.2 Feature Extraction

The first step is feature extraction. Since visual data tends to be extremely high-dimensional (for scale, a low-resolution 160×120 frame contains 19,200 pixels), it is impractical to train models on raw inputs. Therefore, when processing our videos we need to embed each frame into some lower-dimensional space which encodes the information necessary to characterize an activity. Pseudocode for feature extraction can be found in Algorithm 3.

In defining the encoding, we emulate previous approaches (e.g. Kolekar and Dash in [5]) and extract features based on shape and movement (optical flow). Specifically, we define for each frame a vector containing some combination of centroid, edge, and optical flow features.

- **Centroid:** the centroid of thresholded points in the image.
- **Edge:** a window around the centroid in the edge image (dimensionality reduced using PCA).
- **Optical flow:** a frequency representation of dense optical flow values in the primary region of interest. Optical flow is computed using either the Lucas-Kanade method (for tracking points) or Farneback's algorithm (for determining all flow values within a block).
    - We include **divergence**, **curl**, and **average velocity** as supplemental flow features.

3

Algorithm 3 Feature Extraction Algorithm
_____
1: For each frame F in video:
2:      $G \leftarrow$ grayscaled F
3:      Edge image $E \leftarrow$ Canny edge detection on F
4:      Foreground $O \leftarrow F$ minus background
5:      Compute edge features on O
6:      Compute centroid features on O
7:      Compute optical flow features on E_{prev}, E_{curr}
8: Use PCA to post-process edge and optical flow features
9: Combine all features into one vector
_____

3.3 Model Construction

As previously described, we train a first-order left-right hidden Markov model for each activity in our dataset. Given a sequence of observed feature vectors, the Baum-Welch algorithm is used to estimate the parameters of the model. A high-level description of the algorithm can be found in Section 2.3.

3.4 Video Classification

We use maximum likelihood estimation in order to classify the activity. In other words, we compute the log probability of a given video's feature vector under each trained HMM, and choose the HMM associated with the highest probability.

On top of MLE, we do two things. First, we "normalize" the output of each model according to its output statistics over a randomly generated corpus of arbitrary videos. The goal of this step is to make the output distributions of each model resemble each other, so that none have more weight than the others. Second, we borrow a technique from machine learning and ensemble multiple HMMs for each activity. Namely, we train multiple Markov models with varied parameters in a bagging fashion (on different subsamples of the data). As seen in the results section, we find that this provides a slight edge over the vanilla model in most cases.

4 Experiments

We train and evaluate our method on videos from the KTH action dataset. The KTH dataset consists of six hundred 160×120 videos, over which six different activities are performed by 25 different people. Conveniently, the videos take place over a homogeneous background; this simplifies the job of background subtraction, which is not the focus of this work.

The six activities included in the dataset are walking, jogging, running, boxing, handwaving, and handclapping. We train a separate HMM for each activity, and judge our algorithm by its multi-class recognition accuracy.

We split the full dataset into a training set of 384 videos and a test set of 216 videos (equating to 64 training videos and 36 test videos per activity).
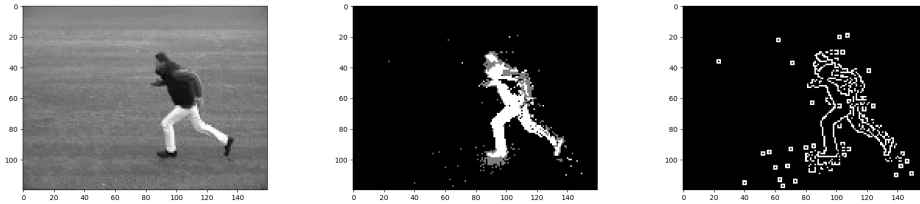

Figure 1: Raw frame, foreground mask, edge image

5 Results

Table 1 contains the classification accuracy associated with each of the activities in the KTH dataset. Each percentage denotes the proportion of correctly classified videos for the activity.

Table 1: ensemble of five GMM HMMs trained with different snapshots of the data

Activity	boxing	handclapping	handwaving	jogging	running	walking
Train %	84%	68%	52%	52%	65%	63%
Test %	81%	72%	75%	72%	69%	58%
Random %	17%	17%	17%	17%	17%	17%

Note: since there are six classes, uniform random sampling would yield a classification accuracy of 17% in expectation. Our method performs quite a bit better than this, showing that it has learned something about each activity.

Also, performance on the test set is similar to performance on the training set, demonstrating to our satisfaction that the classifier can generalize to unseen videos. In a few cases (handwaving and jogging), the model actually does significantly better on the test videos than it does on the training videos. This is potentially due to the test set's smaller size, leading to greater variance in performance (on top of the existing variance from data subsampling during training).

As justification for our use of an ensemble, Table 2 depicts a comparison in classification accuracy between an ensemble-based model and a non-ensemble-based model. The ensemble-based model performs at least as well or better on four of the six activity datasets.

Table 2: comparison against HMM trained on size-25 edge features and size-25 optical flow features

Activity	boxing	handclapping	handwaving	jogging	running	walking
Test % (with ensemble)	81%	72%	75%	72%	69%	58%
Test % (without ensemble)	61%	47%	61%	53%	75%	81%

6 Discussion

Our method does relatively well at classifying videos, and significantly outperforms a baseline random model. However, it is far from the state of the art when it comes to human activity recognition. One limitation is that we're using Markov models, which exhibit a memoryless property in that each state depends only on a previous state, whereas newfangled methods utilize more complex models such as LSTMs (which are able to take into account a long-term memory of a sequence, and perhaps understand more about a behavior that way).

If constrained to the domain of Markov, however, the main limitations likely stem from the feature representations we're using. There are many features that could be included that we have not yet tried, e.g. raw flow values in multiple small windows of greatest magnitude, eddies in the flow field, the DFT of distances from the centroid to human contours... the list goes on. Thus far, we have found the feature representation to make the biggest impact on performance.

Furthermore, we have a large number of parameters to tune (see the configuration file in our codebase [2]) and are not quite sure which ones give the best results. We have found that different parameter settings can sway the performance of our model to a large degree, in one instance misclassifying everything and in another classifying everything to a medium ($\sim$40%) accuracy. Parameters include, e.g., which features to use, dimensionality of feature spaces, number of models in the ensemble, and subsample percentages.

---

[2]`https://github.com/ohjay/hmm_activity_recognition`

7 Conclusion

In this report, we have presented an HMM-based classification approach to human activity recognition. We extend previous approaches with custom-defined feature representations (e.g. PCA ROI edges and histograms of flows), normalization of output distributions, and Markov model ensembles. As alluded to in the previous section, there is much work that could still be done, e.g. re-engineering features and enacting searches over the parameter space for the best values; we haven't yet had much time to experiment. Nevertheless, on the whole we have found Markov models to give unprecedentedly strong results in the challenging task of activity recognition.

References

[1] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? A new model and the kinetics dataset. *CoRR*, abs/1705.07750, 2017.

[2] Fabien Baradel, Christian Wolf, Julien Mille, and Graham W. Taylor. Glimpse clouds: Human activity recognition from unstructured feature points. *CoRR*, abs/1802.07898, 2018.

[3] Song-Mi Lee, Sang Min Yoon, and Heeryon Cho. Human activity recognition from accelerometer data using convolutional neural network. In *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 131–134, Feb 2017.

[4] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: a local svm approach. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 32–36 Vol.3, Aug 2004.

[5] M. H. Kolekar and D. P. Dash. Hidden markov model based human activity recognition using shape and optical flow based features. In *2016 IEEE Region 10 Conference (TENCON)*, pages 393–397, Nov 2016.