Deterministic Initialization of Hidden Markov Models for Human Action Recognition

Zia Moghaddam and Massimo Piccardi
Faculty of Engineering and Information Technology,
University of Technology, Sydney (UTS), Australia
{ziam, massimo}@it.uts.edu.au

Abstract

Human action recognition is often approached in terms of probabilistic models such as the hidden Markov model or other graphical models. When learning such models by way of Expectation-Maximisation algorithms, arbitrary choices must be made for their initial parameters. Often, solutions for the selection of the initial parameters are based on random functions. However, in this paper, we argue that deterministic alternatives are preferable, and propose various methods. Experiments on a video dataset prove that the deterministic initialization is capable of achieving an accuracy that is comparable to or above the average from random initializations and suffers from no deviation thanks to its deterministic nature. The methods proposed naturally extend to be used with other graphical models such as dynamic Bayesian networks and conditional random fields.

1. Introduction

Human action recognition is a very active research area in computer vision with main applications to video surveillance, video retrieval, human-computer interaction and others. Understanding human behavior is a high-level task relying on several, lower-level tasks such as segmentation, tracking and posture recovery. The typical goal of automatic action recognition is the classification of a given image sequence as one of several classes of pre-defined actions.

Many different approaches for action recognition have been proposed over the past two decades. The most recent surveys in [1, 2] offer a good overview. However, it is a common opinion that many open issues still affect the efficacy of action recognition. As a main challenge, the instances of the same action by various people are significantly different; moreover, every individual performs each action in a different manner over various instances, both in space and time. This can be formulated as a problem of high, intrinsic within-class variability. Further, the visual appearance of the individual performing the action varies with the viewpoint and illumination conditions, motivating ongoing research for invariant feature sets. Adding to the challenge, the number of samples available for training is typically limited compared to the parameters, preventing a "brute force" approach.

In terms of recognition approaches, two main lines of investigations have been followed: 1) recognizing the action directly in the time domain; and 2) recognizing the action by probabilistic models. The former group has dynamic time warping (DTW) as its main representative; the latter has the hidden Markov model (HMM). Despite the recent renaissance in interest in time warping approaches, probabilistic models such as HMM have maintained widespread adoption for their recognized strength against the intrinsic variations of action instances. Other probabilistic graphical models such as dynamic Bayesian networks (DBN) and conditional random fields (CRF) have also been used with a significant degree of a success. However, certain problems with the training of probabilistic models are still partially unresolved. The main principle guiding the learning of a model from a set of action samples, $X=\{x_1,..x_N\}$, is to learn its parameters, λ, with maximum likelihood (or maximum a posteriori, wherever prior distributions for the parameters are available). The expression of the likelihood, $p(X|\lambda)$, is typically too complicated to suggest a direct maximization in the parameters and therefore Expectation-Maximization (EM) algorithms have been predominant solutions. However, it is well known that EM algorithms can only find local maxima for the likelihood, and that such maxima strongly depend on the arbitrary initialization made for EM. Moreover, the problem of the quality of the maxima and the generalization to unseen examples is often exacerbated by the scarcity

of training samples. For this reasons, this paper investigates and presents a number of approaches that can improve the effectiveness of model learning from a limited set of samples.

The rest of the paper is organized as follows. Section 2 offers a brief review of the related work. Section 3 summarizes HMM to the extent required by the paper. In Section 4, we present the various HMM initialization strategies, while in Section 5 we present the simple feature set used for the experiments. Experimental results are reported and discussed in section 6. Finally, conclusions are presented.

2. Related work

Using HMM for human action recognition goes a long way back. The first paper that we are aware of, from Yamato *et al*., dates 1992 [3]. The authors used HMM to recognize six different tennis actions. In their work, each frame is background-subtracted and the extracted foreground object is partitioned into a grid of blocks, centred on the centroid. The number of foreground pixels in each block is the feature vector that is then mapped onto a symbol by vector quantization. Discrete-output HMMs with 36 states are used for recognition. This early work already epitomises two major problems of action recognition: a) the adoption of a discriminative and workable feature set and b) the choice of a suitable recognition approach.

For the feature set, a variety of approaches have been exploited, including optical flow [4], body parts tracking [5, 6], silhouettes [7] space-time interest points [8] and local interest points [9, 10]. Researchers are left with the decision whether to use a rich feature vector, possibly invariant to the viewpoint (e.g. [11]), or a simple, fast-to-extract feature vector designed with opportunistic action discrimination in mind. Along the latter lines, Fujiyoshi and Lipton introduced a "star skeleton" method that identifies 5 points with high convex curvature along the silhouette contour; such points represent the top of the head and the extremities of the four limbs [12]. In the approach, the distance between each contour point and the centroid is first calculated to produce a distance function along the curvilinear co-ordinate. The function is then smoothed and five local maxima found by the derivative zero-crossings. Although this feature vector is view dependent, it is fast to extract and low dimensional.

Chen *et al*. in [9] developed the work of [12] and reported that the star skeleton method often achieves incorrect association between maxima and the expected body parts. For this reason, they proposed an adaptive smoothing filter that always detects only and exactly five maxima and relaxed the association of such maxima with physical parts. Also Li and Xu in [10] used the star skeleton feature vector, but introduced posture priors to compensate the observation probabilities of an HMM. In this work, we make use of a simple feature vector showing similarities with the star skeleton, but we enforce anatomical priors restricting the search for maxima to pre-determined angular sectors.

Various graphical models have been used for recognising actions from observation sequences. While the main model has been the HMM, other models such as HMM variants (coupled, hierarchical, layered, entropic etc.), DBNs, CRFs have been used (e.g. [13, 14, 15]). However, they are all highly parametric models and the tuning of their parameters may prove unsatisfactory. In particular, EM learning is sensitive to the choice of the initial parameter assignment, and this problem was recognised and addressed by various authors. For instance, Ferrer *et al*. in [16] reviewed various HMM initialization methods based on random techniques and introduced their own method based on averaging multiple random runs. Unfortunately, random initializations are prone to performance variance. In a recent work, Toledano *et al*. have explored three different ways of initializing HMM training: 1) by a fixed template for all classes; 2) by historical averages; and 3) by oracle initialization (this last only to establish offline upper bounds) [17]. While these methods remove undesired randomness, they are not adaptive in the training samples. Therefore, in this paper we propose various deterministic initialization methods, yet adaptive on the actual training set.

3. Action classification using HMM

Using HMM for action recognition converts the recognition problem into classification of time series. A much-cited tutorial on HMM and its three main problems – evaluation, decoding and estimation – can be found in [18]. Let us call C the set of K action classes, $C = \{C_1,\ldots,C_k,\ldots,C_K\}$. Given an HMM for each class, noted by its set of parameters, λ_k, k=1..K, maximum-likelihood classification of a time series can be achieved as:

$$C_k^* = \arg\max_k \left(p(O \mid \lambda_k)\right) \quad (1)$$

where $p(O|\lambda_k)$ is a likelihood function that can be effectively computed based on the forward or backward algorithm [18]. If full Bayesian

classification is sought, priors and costs can be easily added.

3.1. Hidden Markov model

HMM is a probabilistic graphical model in which the modeled system has observed outputs, or observations, but the states are hidden. The observation sequence is noted as $O=\{o_1,...,o_T\}$, where T is the length of the observation sequence. An HMM with N states is represented by the following parameter set:

$$\lambda = \{A, B, \pi\} \quad (2)$$

where A is the N x N state transition probability matrix, B are the observation probabilities and π are the N x 1 initial state probabilities. In our case, the observations are continuous, multivariate random variables and their distribution in each state is modeled by a mixture of M Gaussian components (Gaussian Mixture Model - GMM):

$$b_j(o) = \sum_{l=1}^{M} c_{jl} G(o \mid \mu_{jl}, \Sigma_{jl}), \quad j = 1...N \quad (3)$$

In (3), μ_{jl} and Σ_{jl} are the mean and covariance of the l-th Gaussian and c_{jl} is its weight in the mixture. Hence, the total size of B is (N * M * sizeof $\{\mu_{jl}, \Sigma_{jl}, c_{jl}\}$). Such a number is typically high and confirms that an HMM is a highly parametric model.

3.2. HMM training

During training, the HMM parameters are estimated to fit the training observation sequences with maximum likelihood [18]. The most popular HMM training algorithm is the Baum-Welch re-estimation algorithm [18], which is of EM style. Like all EM algorithms, it guarantees convergence to a local optimum (or a saddle point) of the data likelihood, and the position and quality of such a maximum depend in turn on the initialization parameters. Moreover, the set of HMM parameters, λ, contains two hyperparameters: the number of states, N, and the number of Gaussian components in each $b_j(o)$, M. For these two parameters, we simply adopt exhaustive search over a plausible range, N, $M \in \{1\ldots5\}$. As software, we have used and extended Kevin Murphy's HMM toolbox for Matlab [19].

4. HMM parameters initialization

The Baum-Welch re-estimation algorithm requires an initial assignment of the HMM parameters to initiate training. While all the parameters influence the outcome of training, in the following we focus only on B because of its typically overwhelming size. For instance, in an HMM with $N = M = 5$, $F = 10$-dimensional observations (a conservative figure) and full covariance matrices, the size of B is equivalent to 1,645 scalar parameters.

The problem with initialization stems from the fact that only a set of training observation sequences is given, without knowledge of the states generating the observations. The training data permit us to easily estimate $p(o)$, the observation probability marginalised over the states; yet, our estimation targets are the conditional observation densities, $b_j(o) := p(o|q=j)$. Before we start describing our initialization approaches, we illustrate a conventional method taking Murphy's toolbox as the reference. Parameter B requires to be initialized with N * M sets of weighted Gaussian components, $\{\mu_{jl}, \Sigma_{jl}, c_{jl}\}$. Murphy's toolbox obtains such values by initially training a single GMM with N * M components and then "dispatching" M components to each state in an arbitrary order. The single GMM is, at its turn, learned with a k-means algorithm whose N * M initial centroids are chosen randomly from the data themselves (strictly speaking, k-means does not assume Gaussian distributions for the clusters, but we will treat it as such hereafter). While this procedure can produce effective initialization, it might have to be applied several times before satisfactory parameters can be found. Conversely, all the methods that we propose in the following provide deterministic initialization grounded in the data. They develop over two separate steps (Fig. 1):

1. *Cluster creation*: The first step provides N * M Gaussian clusters as its output. The input consists of N * M chosen centroids and learning is provided by a k-means algorithm.

2. *Cluster dispatching*: The N * M clusters are dispatched over the N states (M clusters to each state).

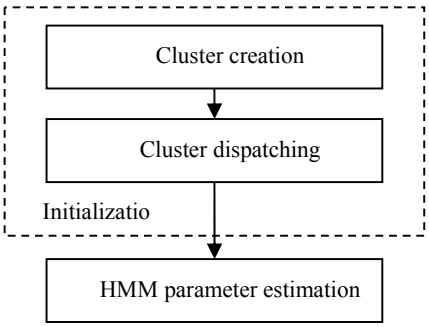

Figure 1. The main steps of HMM learning.

4.1. Cluster creation methods

As mentioned above, we train the GMM by a *k*-means algorithm. This algorithm requires a set of initial centres for each cluster (*initCentres*). In order to devise a practicable deterministic strategy, we proceed as follows: we consider each training sequence $\{O_e\}_{e=1..E}$, with length T_e, and we divide it into $(N*M + 1)$ consecutive segments $\{S_{ep}\}_{p=1..(N*M+1)}$, each of $\lfloor T_e/N*M \rfloor$ length; then, we collect their boundary points as $\{M_{ep}\}_{p=1..N*M}$.

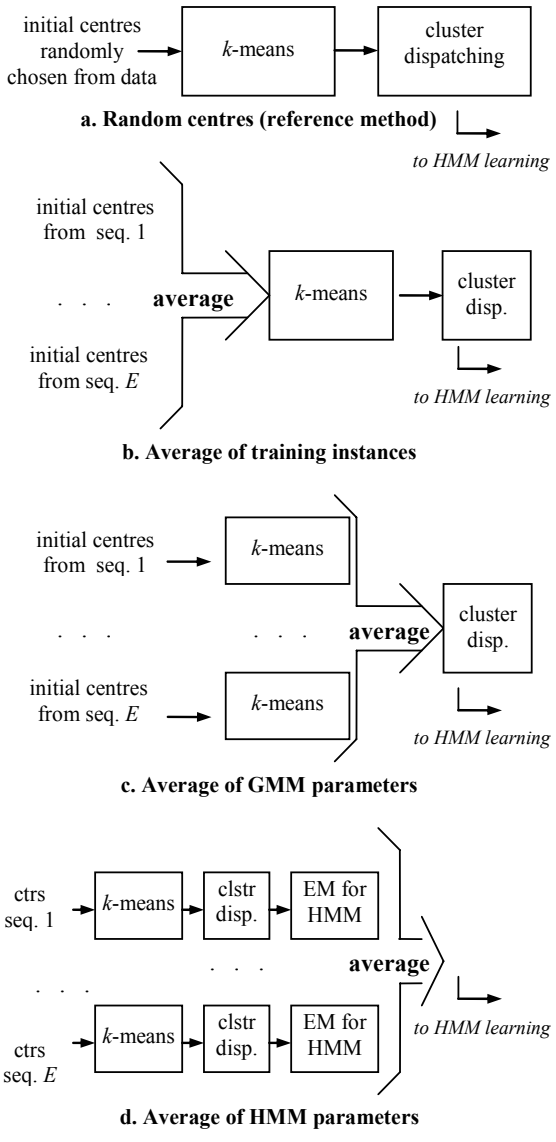

Figure 2. The cluster creation methods.

The aim of this procedure is to sample the training sequence along the time in order to extract sensible starting points for the formation of the observations' clusters and, in turn, the $b_j(o)$ densities. While the dynamic of the human action is certainly not linear in general, this procedure provides useful starting values; the discovery of the non-linearities is the task of the following HMM training stage. As the second requirement for the GMM training, we need to specify the set of samples (*observData*). We articulate our choices of *initCentres* and *observData* as follows:

1. *Random centres (reference method)*: *initCentres* are chosen randomly and the observations from all the training data sequences are coalesced into a single *observData* "supervector" (Fig 2.a).

2. *Average of training instances*: *initCentres*, M_{ep}, are calculated as described at the beginning of this section for each training data sequence and then averaged over all the instances; *observData* is the supervector (Fig. 2.b).

3. *Average of GMM parameters*: Each training data instance is separately used for *initCentres* calculation and as *observData*. The parameters in output are then averaged before cluster dispatching (Fig. 2.c). We care to note that this is an empirical procedure as GMM parameters, and in particular the covariance parameters, are not linear. However, we can assume that the various trained models are not too different from one another, and that their linear combination is an acceptable approximation.

4. *Average of HMM parameters*: Each training data instance is separately used for *initCentres* calculation and as *observData*. Parameters are dispatched and used to train individual HMMs. The trained HMM parameters are averaged (again, under a small-signal linear assumption) and such averages are used as the input for the final HMM learning (Fig. 2.d).

4.2. Cluster dispatching methods

The clusters created during the previous stage need then to be "dispatched" as modes of the observation distributions of the HMM states. This action may be regarded as non critical since it may appear that changes to the modes' assignments will be compensated by corresponding changes to A, the state transition probabilities. However, this cannot account for the different constructive interference of modes. We propose two cluster dispatching methods contrasting them to the reference method:

1. *Appearance order (reference method)*: Dispatching clusters based on their appearance order

in the set is the simplest way to proceed, but completely arbitrary.

2. *Nearest neighbours:* The goal of this method is to put clusters with the mutually closest centres in the same state. First, we compute all the Euclidean distances between pairs of clusters' centres. Then, we create all the possible partitions of clusters onto states and for each partition we compute its corresponding overall Euclidean distance. Finally, the partition with minimum overall Euclidean distance is selected as the best dispatching. The total number of possible partitions, T_P, for an N-state HMM with M modes per state is given by:

$$T_P = \frac{\binom{M*N}{M} * \binom{M*(N-1)}{M} * \ldots * \binom{M}{M}}{N!}, \quad (4)$$

$$\text{where } \binom{n}{k} = \frac{n!}{k!*(n-k)!}, \quad (5)$$

As can be seen, T_P is, unfortunately, very high and the combinatorial exploration proves extremely time-consuming even for reasonably low values of N and M.

3. *Feature sorting:* The goal of this method is to approximate the nearest-neighbour dispatching with a much lower computational load. The clusters' centres are F-dimensional vectors: therefore, they can be seen as an F x (N * M) matrix. Here, each row is first sorted in value order and the ranking of each cell in the sorted row retained. Then, the average of the ranks along each column is used to determine the global rank of each cluster. Clusters are eventually dispatched to states in global ranking order. This method has a favourable $O(n \log n)$ complexity in *(M * N)*.

5. Sector extreme points as feature set

The various methods proposed for cluster creation and dispatching provides a deterministic approach to model learning. A complementary aspect to the choice of the model is the feature vector to adopt. Given the tight real-time constraints of video surveillance, we chose to extract a minimal set of shape descriptors with the following procedure: we first extract the human silhouette from the background and divide it into five circular sectors centred around the silhouette's centroid. Then, for each sector we determine the silhouette's contour point farthest from the centroid. We call the resulting five points 'sector extreme points' and we assume that they would be in frequent correspondence with anatomical points. While this is not meant as an exact tracking procedure, the trajectory of these five points proves action-discriminative. Fig. 3 shows an example of sector extreme point trajectories. Further, to also encode the absolute position of the object, we add the centroid's coordinates to the feature vector.

5.1. Feature processing by standardisation

Many of the values in the feature vector are affected by the anthropometry of the subjects i.e. their height and limbs' length. In some cases, the feature values for different subjects would be in totally different ranges and cause over-estimates of the observations' covariances when learning from multiple subjects. In turn, this would affect the classification accuracy. One way to address this problem is to normalise the feature values by common preprocessing techniques, such as standardisation or whitening. For this work, we decided to standardise the observations over the sequence they belong to as:

$$O^s_{j,t} = \frac{O_{j,t} - \mu(O_j)}{\sigma(O_j)} \quad j = 1...F, t = 1...T \quad (6)$$

6. Experiments

We have tested our action recognition approach with the popular Weizmann human action video dataset [7]. The dataset includes 10 actions (Run, Walk, Skip, Bend, Side, Jack, Jumping Jack, Jump, Pjump, Wave1 and Wave2) performed by 9 different subjects and the videos are stored in low resolution AVI format (180*144 pixels). Alongside the original videos, the dataset also includes the silhouettes (median background subtraction) of the video sequences. While this dataset is of relatively small size (93 samples) and somehow simplistic, we decided to use it in this work as it allows direct comparison with most of the literature. As validation, we used "leave-one-out" cross validation i.e. in each run we leave one subject out during training and we use it for testing. The final accuracy result is the average of the nine runs over the various subjects.

6.1. Experiments on feature processing

We ran the first set of experiments to find the influence of feature processing. Table 1 compares the results between the original features and the standardised features using *Average of GMM*

parameters for cluster creation and *Appearance order* for cluster dispatching.

Table 1: Classification accuracy (%) with the original and the standardised features.

Original features

	M=1	M=2	M=3	M=4	M=5
N=1	82.8	75.3	73.1	72	72
N=2	74.2	74.2	74.2	73.1	72
N=3	73.1	73.1	73.1	71	69.9
N=4	73.1	72	68.8	68.8	59.1
N=5	73.1	67.7	66.7	59.1	63.4

Standardised features

	M=1	M=2	M=3	M=4	M=5
N=1	94.6	92.5	92.5	91.4	93.5
N=2	94.6	91.4	93.5	94.6	90.3
N=3	94.6	93.5	86	93.5	81.7
N=4	89.2	91.4	92.5	78.5	52.7
N=5	88.2	90.3	82.8	51.6	46.2

The results clearly show the effectiveness of applying standardization to the feature set. Using the other methods for cluster creation and cluster dispatching confirmed this conclusion. Hence, for the following experiments we used the standardised feature set.

6.2. Experiments on cluster creation methods

In the second set of experiments, we compared our various cluster creation methods with the reference method. Here, the cluster dispatching needs to be fixed to one of the proposed methods discussed in section 4.2; we chose *Appearance order* for direct comparison with the reference method. Table 2 reports the classification accuracy. For *Random centres,* we report the average alongside the standard deviation over 6 different runs. The main problem with the *random centres* method is that it might have to be applied several times before satisfactory parameters can be found. Conversely, two of our methods obtain an accuracy that is mildly higher than the average of the random runs, and all have zero deviation since they are deterministic. The *Average of training instances* and the *Average of HMM parameters* achieve the highest accuracy. However, the latter reports very low accuracy for high values of $N * M$, probably because the linear approximation becomes more tenuous. By repeating the experiment with the other dispatching methods, we achieved equivalent results.

Table 2: Classification accuracy (%) with the different cluster creation methods.

Random centres (reference method)

	M=1	M=2	M=3	M=4	M=5
N=1	94.6	92.3±1.6	92.5±1.7	93.7±2.9	93.0±1.9
N=2	93.2±1.1	93.4±2.0	92.5±2.3	93.0±2.2	90.9±1.8
N=3	95.2±1.1	93.9±1.1	92.1±2.2	92.3±0.8	92.5±1.9
N=4	92.3±2.1	90.1±1.7	90.5±2.0	90.7±1.6	89.2±0.7
N=5	91.8±1.9	90.9±1.3	90.0±1.8	89.4±1.6	89.6±1.6

Average of training instances

	M=1	M=2	M=3	M=4	M=5
N=1	94.6	91.4	89.2	93.5	92.5
N=2	93.5	93.5	94.6	91.4	90.3
N=3	95.7	91.4	92.5	89.2	92.5
N=4	90.3	93.5	89.2	92.5	89.2
N=5	92.5	87.1	88.2	88.2	83.9

Average of GMM parameters

	M=1	M=2	M=3	M=4	M=5
N=1	94.6	92.5	92.5	91.4	93.5
N=2	94.6	91.4	93.5	94.6	90.3
N=3	94.6	93.5	86	93.5	81.7
N=4	89.2	91.4	92.5	78.5	52.7
N=5	88.2	90.3	82.8	51.6	46.2

Average of HMM parameters

	M=1	M=2	M=3	M=4	M=5
N=1	94.6	90.3	93.5	95.7	94.6
N=2	93.5	93.5	91.4	92.5	91.4
N=3	92.5	94.6	89.2	94.6	87.1
N=4	92.5	95.7	93.5	82.8	52.7
N=5	90.3	92.5	86	52.7	48.4

6.3. Experiments on cluster dispatching methods

The third experiment was designed to explore the best cluster dispatching method among those described in section 4.2. In this experiment, we adopted the cluster creation method that reported the best performance in the previous experiment (*Average of training instances*). The achieved accuracies using different dispatching methods are shown in Table 3. Cases $N = 1$, $M = 1$ are not reported as they are not significant. The *Nearest neighbours* method proved very time-consuming and we were not able to complete the tests for $N * M > 16$ in reasonable time. The *Feature sorting* method

seemed to provide the best tradeoff between speed and accuracy.

Table 3: Classification accuracy (%) with the different cluster dispatching methods.

Appearance order (reference method)

	M=2	M=3	M=4	M=5
N=2	93.5	94.6	91.4	90.3
N=3	91.4	92.5	89.2	92.5
N=4	93.5	89.2	92.5	89.2
N=5	87.1	88.2	88.2	83.9

Feature sorting

	M=2	M=3	M=4	M=5
N=2	94.6	95.7	94.6	94.6
N=3	91.4	91.4	90.3	91.4
N=4	93.5	87.1	94.6	89.2
N=5	94.6	90.3	86	84.9

Nearest neighbours

	M=2	M=3	M=4	M=5
N=2	92.5	93.5	94.6	95.7
N=3	94.6	90.3	88.2	92.5
N=4	91.4	87.1	94.6	Not tested
N=5	92.5	91.4	Not tested	Not tested

Table 4: Confusion matrix (*N*=2, *M*=3, Average of training instances, Feature sorting).

	Class relative accuracy									
	Bend	Jack	Pjump	Jump	Run	Side	Skip	Walk	Wave1	Wave2
Bend	1	0	0	0	0	0	0	0	0	0
Jack	0	1	0	0	0	0	0	0	0	0
Pjump	0	0	1	0	0	0	0	0	0	0
Jump	0	0	0	0.78	0	0	0.22	0	0	0
Run	0	0	0	0	1	0	0	0	0	0
Side	0	0	0	0.11	0	0.89	0	0	0	0
Skip	0	0	0	0.10	0	0	0.90	0	0	0
Walk	0	0	0	0	0	0	0	1	0	0
Wave1	0	0	0	0	0	0	0	0	1	0
Wave2	0	0	0	0	0	0	0	0	0	1

Overall, the highest accuracy we achieved across the various initialization methods is 95.7%. Table 4 depicts the full confusion matrix for this case (rows are the ground truth and columns the classification results), showing that the few errors occur mainly between the self-similar classes *Jump* and *Skip*.

Other papers in the literature have reported higher accuracy on this dataset (e.g. 100% [20], 97.8% [15]); however, their approaches are not HMM-based. Using randomly initialized HMMs and a comparable feature set, the best result is from Li and Xu with an accuracy of 92.5% [10].

7. Conclusions

In this paper, we have proposed various deterministic methods for the initial assignment of parameters in HMM learning of human actions. The approaches we proposed ground the choice of the initial parameters in the training data, hoping to permit greater accuracy for the learned model. In Section 6.2, we showed that the deterministic initialization is capable of achieving an accuracy that is comparable to or above the average from random initializations. At the same time, the deterministic approach incurs no deviation over different runs. We argue that the proposed approach can be usefully extended to other discrete state-space models popular for action recognition such as DBNs and CRFs where the probability of observable random variables must be modelled conditional to discrete states.

8. Acknowledgement

This research was partially supported by SenSen Networks Pty Ltd.

9. References

[1] T.B. Moeslund, A. Hilton and V. Kru¨ger, "A Survey of Advances in Vision-Based Human Motion Capture and Analysis," *Computer Vision and Image Understanding*, vol. 104, no. 2-3, 2006, pp. 90-126.

[2] P. Turaga, R. Chellappa, V.S. Subrahmanian and O. Udrea, "Machine Recognition of Human Activities: A Survey," *IEEE Trans. Circ. Syst. for Video Technology*, vol. 18, no. 11, 2008, pp. 1473-1488.

[3] J. Yamato, J. Ohya and K. Ishii, "Recognizing Human Action in Time-Sequential Images using Hidden Markov Model," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 1992, pp. 379-385.

[4] A.A. Efros, A.C. Berg, G. Mori and J. Malik, "Recognizing action at a distance," *Proc. of the 9th IEEE Intl. Conf. on Computer Vision (ICCV)*, 2003, pp. 726-733.

[5] Z.L. Husz, A.M. Wallace and P.R. Green, "Human Activity Recognition with Action Primitives," *Proc. of*

the IEEE Intl. Conf. on Advanced Video and Signal Based Surveillance (AVSS), 2007, pp. 330-335.

[6] Y. Song, L. Goncalves and P. Perona, "Unsupervised learning of human motion," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 25, no. 7, 2003, pp. 814-827.

[7] M. Blank, L. Gorelick, E. Shechtman, M. Irani and R. Basri, "Actions as Space-Time Shapes," *Proc. of the 10th IEEE Intl. Conf. on Computer Vision (ICCV)*, 2005, pp. 1395-1402.

[8] P. Dollar, V. Rabaud, G. Cottrell and S. Belongie, "Behavior recognition via sparse spatio-temporal features," *Proc. of the 2nd Joint IEEE Intl. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (PETS)*, 2005, pp. 65-72.

[9] H.-S. Chen, H.-T. Chen, Y.-W. Chen and S.-Y. Lee, "Human action recognition using star skeleton," *Proc. of the 4th ACM Intl. Workshop on Video Surveillance and Sensor Networks (VSSN)*, ACM, 2006.

[10] N. Li and D. Xu, "Action recognition using weighted three-state Hidden Markov Model," *Proc. of the 9th Intl. Conf. on Signal Processing (ICSP)*, 2008, pp. 1428-1431.

[11] Y. Shen and H. Foroosh, "View-invariant action recognition using fundamental ratios," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1-6.

[12] H. Fujiyoshi and A.J. Lipton, "Real-time human motion analysis by image skeletonization," *Proc. of the 4th IEEE Workshop on Applications of Computer Vision (WACV)*, 1998, pp. 15-21.

[13] J. K. Aggarwal and Q. Cai, "Human Motion Analysis: A Review," *Computer Vision and Image Understanding*, vol. 73, no. 3, 1999, pp. 428-440.

[14] L. Wang, W. Hu, and T. Tan, "Recent developments in human motion analysis," *Pattern Recognition*, vol. 36, no. 3, 2003, pp. 585-601.

[15] L. Wang and D. Suter, "Recognizing Human Activities from Silhouettes: Motion Subspace and Factorial Discriminative Graphical Model," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1-8.

[16] M.A. Ferrer, I.G. Alonso and C.M. Travieso, "Influence of initialisation and stop criteria on HMM based recognisers," *Electronics Letters*, vol. 36, no. 13, 2000, pp. 1165-1166.

[17] D. T. Toledano, J. G. Villardebo, L. H. Gomez, "Initialization, training, and context-dependency in HMM-based formant tracking," *IEEE Trans. on Audio, Speech, and Language Processing*, vol. 14, no. 2, 2006. pp. 511-523.

[18] L.R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proceedings of the IEEE*, vol. 77, no. 2, 1989, pp. 257-286.

[19] K. Murphy, "Hidden Markov Model (HMM) Toolbox for Matlab," 1998 (last updated in 2005); http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html.

[20] A. Fathi and G. Mori, "Action recognition by learning mid-level motion features," *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008, pp. 1-8.

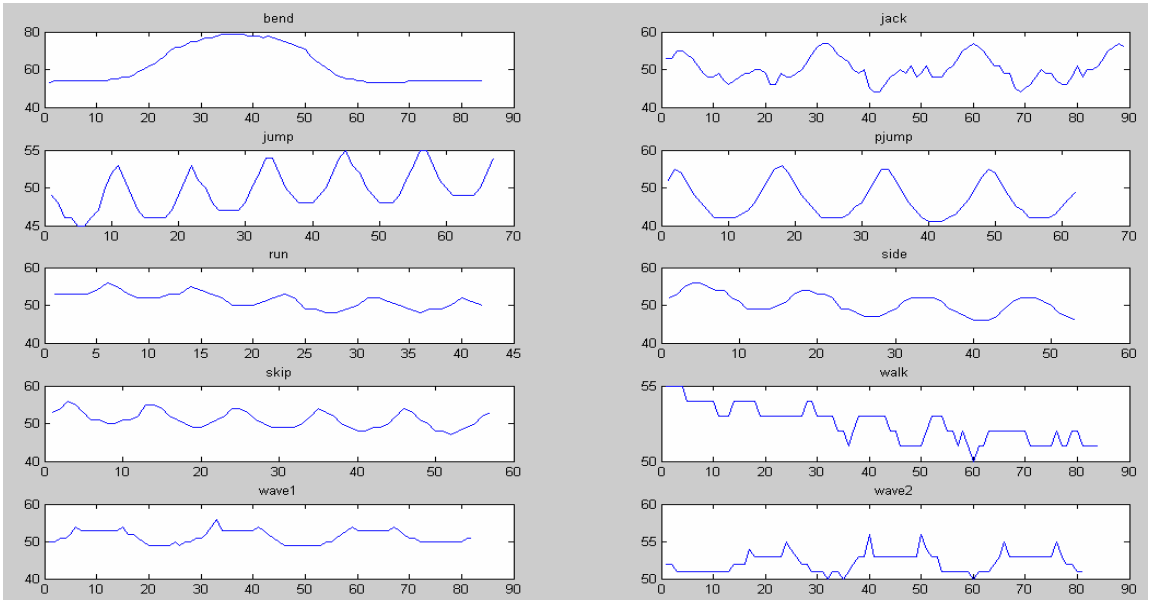

Figure 3. The values of Head (Row) feature for 10 actions performed by one subject in the Weizmann dataset.