Comparison between Genetic Algorithms and Baum-Welch algorithm for the adjustment of HMM to Classify Human Activities[1]

Óscar Pérez[1,2], Massimo Piccardi[1], Jesús García[2] , Miguel A. Patricio[2], José M. Molina[2]

[1] Faculty of Information Technology
University of Technology, Sydney
{oscapc, massimo}@it.uts.edu.au

[2]Computer Science Department-GIAA
Universidad Carlos III de Madrid
Colmenarejo, Spain
{opconcha,jgherrer,mpatrici}@inf.uc3m.es,
molina@ia.uc3m.es

Abstract. A Hidden Markov Model (HMM) is used as an efficient and robust technique for human activities classification. The HMM evaluates a set of video recordings to classify each scene as a function of the future, actual and previous scenes. The probabilities of transition between states of the HMM and the observation model should be adjusted in order to obtain a correct classification. In this work, these matrixes are estimated using the well known Baum-Welch algorithm that is based on the definition of the real observations as a mixture of two Gaussians for each state. The application of the GA follows the same principle but the optimization is carried out considering the classification. In this case, GA optimizes the Gaussian parameters considering as a fitness function the results of the classification application. Results show the improvement of GA techniques for human activities recognition.

Keywords: Human Activities Classification, Hidden Markov Model, Genetic Algorithms, Expectation-Maximization Algorithm.

1 Introduction

The problem addressed in this paper is the recognition of human activities by analysis of features extracted from a video tracking process, with a special emphasis on data modelling and classifier performance. Categorization of human activity is an active research area in computer vision and covers several areas such as detection, motion analysis, pattern recognition, etc. The usual tasks involved are people detection, identification, motion-based tracking, and activity classification. The final goal is to cover the gap between the low-level tracking information provided by the

[1]Funded by projects CICYT TSI2005-07344, CICYT TEC2005-07-186 and CAM MADRINET S-0505/TIC/0255

system and the interpretation necessary to have an abstract description of the scene. This capability is needed as a practical means to avoid the human burden on operator continuously monitoring video cameras.

Typically, in order to recognize human behaviours, the detection of moving objects is not enough, but an analysis of trajectory and interaction with the scene context (other objects and elements) is necessary to infer the behaviour type [1-2]. Therefore, an important requirement of these systems is the capability to track and maintain identity of all detected objects over time.

The process basically involves matching a measured sequence of observations to a certain set of states (or sequences of states) representing predefined actions. There are several approaches for matching time-varying data. Dynamic time warping (DTW) [3, 4], HMM (hidden Markov models) [5-7], Bayesian networks [8, 9] and declarative models [10]. Human actions are represented by sequences of states which must be inferred from features extracted from images (shapes, body parts, poses, etc.) and attributes extracted from motion such as velocity, trajectory, etc. In this work, three states are used corresponding to INACTIVE, WALKING and RUNNING activities. The features selected to classify these actions are the velocities measured in the scene.

The first step of the application is the learning task. The main problem associated with HMMs is to take a sequence of observations (the speed measurements mentioned above), known to represent a set of hidden states, and fit the most probable HMM [11]; that is, determine the parameters that most probably describe what occurs in the scene. The forward-backward algorithm (Baum-Welch, which is an expectation maximization (EM) algorithm) is used when the HMM parameters are not directly (empirically) measurable, as is very often the case in real applications. The second step is the decoding task, finding the most probable sequence of hidden states given some observations that is, finding the hidden states that generated the observed output. The sequence of states obtained by the Viterbi algorithm is then compared with the ground truth to measure the accuracy.

In this work, we propose a Genetic Algorithm to adjust the HMM matrixes. The fitness function is the classification performance of the second step. We later compare its accuracy against that obtained by the Baum-Welch algorithm for the same task. We prove that GA outperforms the traditional Baum-Welch in the estimation of the parameters, resulting in higher accuracy in the state decoding. In our experiments, the GA achieved 91.11% accuracy while the EM 74.32 %.

In the next section, the HMM structure and Viterbi algorithm are outlined as selected framework to infer the most probable sequence of states according to observations. Section 3 focuses on free parameters of the model and alternative learning techniques with the EM and GA approaches. The fourth section adjusts the parameters of the HMM by means of the two algorithms, and the results are discussed in the fifth section.

2 Hidden Markov Model

The Hidden Markov Model (HMM) [11] is a model framework with state transitions and observation distributions used to infer the sequence of states from the available features extracted from images. It is defined by the triple $\lambda = (A, B, \Pi)$. A are called the state transition probabilities and express the Markovian hypothesis that the value, i, of the current state, S_k, is only dependent on the value, j, of the previous state, S_{k-1}. B are called the observation probabilities, quantifying the probability of observing the O_k value when the current state is j. Eventually, π are called the initial state probabilities and quantify the probabilities of values for the initial state. The actual history of the human's activity states versus time is retrieved using the Viterbi algorithm. They definition and process are characterized by the following elements:

- The target locations as well as the sensor observations are modelled by a set of states, $S = \{S_1, S_2, \cdots, S_N\}$.
- The set of time-invariant transition probabilities

$$a_{ij} = P[q_k = S_j \mid q_{k-1} = S_i], \quad 1 \le i, j \le N \tag{3}$$

where q_k denotes the actual state at time k ($1 \le k \le T$)

- The observation probability distribution

$$b_i(O_k) = P[O_k \mid q_k = S_i], \quad 1 \le i \le N \tag{4}$$

where O_k denotes the observation state at time k.

- The initial state distribution

$$\pi_i = P[q_1 = S_i], \qquad 1 \le i \le N \tag{5}$$

The Viterbi algorithm is then used to find the best state sequence, $q_1^*, q_2^*, \cdots, q_T^*$, given the observation sequence $O_1, O_2, \cdots, O_T$. The highest probability along a single path, which accounts for the first k observations and ends in S_i at time k, is defined as:

$$\delta_k(i) = \max_{q_1, q_2 \cdots, q_{k-1}} P[q_1 q_2 \cdots q_k = S_i, O_1 O_2 \cdots O_k \mid \lambda]. \tag{6}$$

It can be induced that

$$\delta_{k+1}(j) = \max_i [a_{ij} \delta_k(i)] \cdot b_j(O_{k+1}) \tag{7}$$

The best state sequence can be retrieved by keeping track of the argument that maximizes previous equation for each k and j. The complete procedure can be described as follows:

1) Initialization:

$$\delta_1(i) = \pi_i b_i(O_1), \quad 1 \le i \le N \tag{8}$$

$$\varphi_1(i) = 0 \tag{9}$$

2) Recursion:

$$\delta_k(j) = \max_{1 \le i \le N}[\delta_{k-1}(i)\, a_{ij}]\, b_j(O_k),\ 2 \le k \le T, 1 \le j \le N \tag{10}$$

$$\varphi_k(j) = \arg\max_{1 \le i \le N}[\delta_{k-1}(i)\, a_{ij}],\ 2 \le k \le T, 1 \le j \le N. \tag{11}$$

3) Termination:

$$q_T^* = \arg\max_{1 \le i \le N}\left[\delta_T(i)\right]. \tag{12}$$

4) State sequence backtracking:

$$q_k^* = \varphi_{k+1}(q_{k+1}^*), \quad k = T-1, T-2, \cdots, 1. \tag{13}$$

The actual state sequence is usually unavailable and the sensor measurement sequence is thus used instead. The observation probability distribution, $b_j(O_k)$, is a Gaussian mixture likelihood function as mentioned above.

Figure 1 shows the general sketch of what an HMM is.

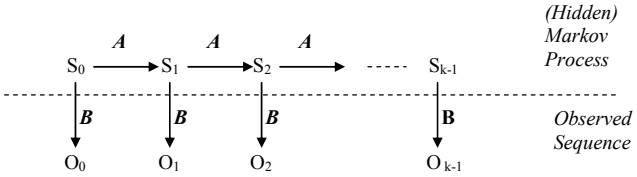

Fig. 1. HMM general scheme.

The model is fully described by the set of parameters $\lambda = \{A, B, \pi\}$.

3 Comparison between the Baum-Welch algorithm and GA.

The well-known Baum-Welch algorithm can be used to simultaneously estimate the state transition probabilities and the observation probabilities in a maximum likelihood framework from the sequence of observations. The states in HMM are assume as discrete values. However, the observations can have either discrete

(limited) values, modelled with probabilities in matrix B, or continuous (or continuous discretised), modelled by conditional probability density functions. In the latter case, HMMs can use MoGs to model their observation probabilities. Each state's value, $i = 1..N$, has its own MoG, independent from those of the other values. Thus, for example, the observation likelihood for i-th state, $b_i(O_k)$, is given by the following MoG:

$$b_i(O_k) = \sum_{l=1}^{M} \alpha_{il} G(O_k, \mu_{il}, \sigma_{il}^2) \tag{14}$$

Then, we can define $p(l \mid O_k, \Theta)$ as the probability of the l-th Gaussian component at sample O_k :

$$p(l \mid O_k, \Theta) = \frac{\alpha_{il} G(O_k, \mu_{il}, \sigma_{il}^2)}{\sum_{p=1}^{M} \alpha_{ip} G(O_k, \mu_{ip}, \sigma_{ip}^2)} . \tag{15}$$

The weights, means and variances ($\alpha_{il}, \mu_{il}, \sigma_{il}^2$) of the observation distributions (the B "matrix"), are then calculated over the set of observed values, O_k, $k = 1..T$. The numerators and denominators are modulated (i.e. multiplied) by the probability of being in state i at time k, $\gamma_i(k)$

$$\alpha_{il}^{new} = \frac{\sum_{t=1}^{T} p(l \mid O_k, \Theta)\, \gamma_i(k)}{\sum_{k=1}^{T} \gamma_i(k)} \tag{16}$$

$$\mu_{il}^{new} = \frac{\sum_{k=1}^{T} O_k \; p(l \mid O_k, \Theta)\, \gamma_i(k)}{\sum_{k=1}^{T} p(l \mid O_k, \Theta)\, \gamma_i(k)} \tag{17}$$

$$\sigma_{il}^{2\,new} = \frac{\sum_{k=1}^{T} (O_k - \mu_{lt})^2 \; p(l \mid O_k, \Theta)\, \gamma_i(k)}{\sum_{k=1}^{T} p(l \mid O_k, \Theta)\, \gamma_i(k)} \tag{18}$$

Finally, the problem could be stated as: How to calculate these three parameters?. In the first case, using an EM algorithm [12], we calculate μ, σ and α, but some problems appear:

1. This algorithm is very sensitive to the initialisation of the parameters

2. We always get a local optimum.
3. The algorithm does not use the ground truth data.

In the second case, we use a GA to calculate the μ, σ and α parameters. This algorithm is not so very sensitive to the initialisation of the parameters, because GA begins with many individuals and mutation operator generates new individuals in different search zones. This characteristic allows to the GA to avoid getting the local optimum and hopefully finding a global optimum. Moreover, during the training process we know exactly how to match the output of the algorithm with the ground truth and use this as the fitness function.

4 Methodology and Results

The aim of our experiments was the classification of human activities by means of a HMM, carrying out a comparison between the Baum-Welch and Genetic Algorithm used for the adjustment of the parameters of the HMM. Moreover, we desired to use the minimum number of features for the easy and fast performance of the HMM.

We used the CAVIAR video dataset [13] and selected two long videos in which several actors behaved in different ways: "Two people meet, fight and run away" and "Two people meet, fight, one down, other runs away". The ground truth was taken from the PETS 2004 website [14] since the format was more convenient for our experiments. Then, the ground truth was adapted in the way that we only chose the identification and position of each actor.

Among all the activities showed in these videos we reduced the set to three: Inactive ("in"), Walking ("wk") and Running ("r").

The ground truth was found by hand-labelling and we must take into account for the final results the subjectivity when classifying, especially between the classes walking and running. Therefore, it is important to highlight that the ground truth is made up of sequences that show the activity of each actor from the point of their appearance until they disappear.

Finally, the whole programming was developed in Matlab.

The features selected to classify the actions are the velocities measured from real cases. The velocities are calculated considering the position of the human in different frames. In our particular case these velocities are calculated following these equations:

$$speed_5 = \frac{1}{5}\sqrt{(x_i - x_{i-5})^2 + (y_i - y_{i-5})^2} \tag{1}$$

$$speed_{25} = \frac{1}{25}\sqrt{(x_i - x_{i-25})^2 + (y_i - y_{i-25})^2} \tag{2}$$

where x_i and y_i are the position of the people in each frame *i*.

The next figure shows how the data are distributed according to the two velocities:

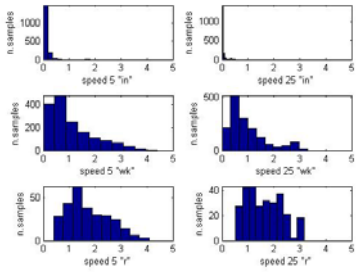

Figure 2. Distribution of the data according to speed_5 and speed_25 for the classes "in" (inactive), "wk" (walking) and "r" (running)

Thus, the first experiments were the adjustment of the HMM parameters using the Baum-Welch algorithm and the Kevin Murphy Matlab toolbox [15]. The prior probability is fixed to be [state 1=1; state 2= 0; state 3 = 0], which means that we assume that the first state is always inactive. This assignment results very useful when decoding the sequence with Viterbi, as we do not know a priori the correspondence between the states in the Viterbi output and those in the ground truth. By fixing this probability we assure that the first code of the Viterbi output will match the inactive state.

The initialization of the variables for the EM algorithm influences greatly in the final result. Hence, although the first option was to initiate randomly the different variables, the poor results obtained made us initialise the variables by taking into account the primitive knowledge of the observation data. Therefore, for our experiments (three states and three Gaussians per state) the initialisation of the 51 variables was fixed like as such:

- Means (12 parameters) and covariances (24 parameteres): μ_i = [0.1; 2.0; 5.0] $\sigma_i = 1$
- Weight for each of the Gaussians (6 parameters): random
- State transition matrix (9 parameters): a_{ij} =[0.8 0.19 0.01; 0.1 0.8 0.1; 0.01 0.8 0.19] where the transition from inactive to running (and vice versa) is very small.
- Maximum number of iterations for the EM algorithm: *max-iter=50*

The result of the EM algorithm gave the State transition matrix and the Observation probability matrix that define the HMM. Then, we use the Viterbi algorithm to determine the most probable sequence of hidden states given a sequence of observations as input of our HMM and measure accuracy against the ground truth.

The second set of experiments adjust the parameters of the HMM by means of a Genetic Algorithm and the Matlab Genetic Algorithm Toolbox of the University of Sheffield [16]. The individual is made up of the 51 parameters mentioned above.

Again, the prior probability is fixed to [state 1=1; state 2= 0; state 3 = 0] in order to have the same conditions for both experiments.

Nevertheless, the initiation of the individuals is carried out randomly since the GA is not as sensitive to the initiation as the EM algorithm was. Other important parameters of the GA are fixed as below:

- No. of individuals: NIND = 45;
- Generation gap: GGAP = 0.8 which means that NIND x GGAP=36 are produced at each generation.
- Crossover rate: XOVR = 1
- Mutation rate: MUTR = 1/(41);
- Maximum no. of generations: MAXGEN = 500;
- Insertion rate: INSR = 0.9 that specifies that 90% of the individuals produced at each generation are reinserted into the population.

Each individual of the population is validated with the Viterbi algorithm. As a result, the fitness function is defined as the comparison between the output of the Viterbi algorithm and the ground truth.

The data are divided into two big groups of sequences (Holdout cross-validation): one of 8 sequences of 2478 data in total and another one of 5 sequences and 1102 data. The first one is used to train while the second validates the learned HMM. The next table shows the total error for the training and validation for the EM and the GA.

Table 1. Total Error of classification for the training and validation data.

Total Error (%)	**Training**	**Validation**
EM	20.82	25.68
GA	10.17	8.89

The next matrix show the confusion matrix for the EM and the GA where each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. Hence, the first row is the real class one or inactive, the second row the real class for walking and the third row stands for the class running. Subsequently, the first column represents the number of samples classified as inactive and the second and third column the ones classified as walking and running respectively.

Table 2. Confusion matrix for the classification with the EM and GA algorithms for the validation data.

EM	Predicted			
Actual		in	wk	r
	in	550	1	0
	wk	12	179	269
	r	0	1	90

GA	Predicted			
Actual		in	wk	r
	in	551	0	0
	wk	6	453	1
	r	0	91	0

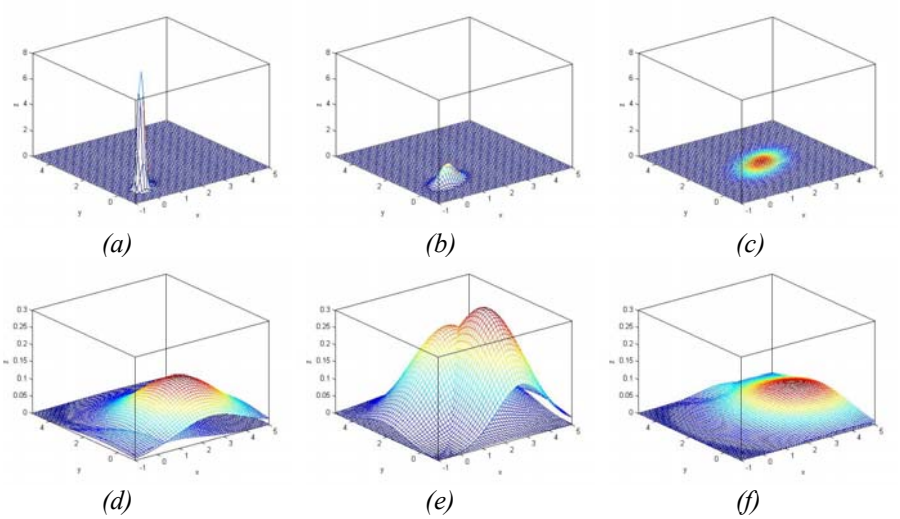

Figure 3. Mixture of Gaussians derived from the EM algorithm (in a scale 0-8) for inactive (a), walking (b) and running (c). Mixture of Gaussians derived from the GA algorithm (in a scale 0-0.3) for inactive (d), walking (e) and running (f).

5 Discussion and future works

The results show a clear better performance for the GA over the EM algorithm. Furthermore, the EM algorithm does not take advantage of having two Gaussian distributions per state and merges both functions in only one.

The EM inferred a Gaussian that classifies 90 out 91 running samples in a correct way, but paying the high price of misclassifying 269 walking samples as running. Nevertheless, the genetic algorithm enhances the total classification by ignoring the running samples and classified in a cored way 453 walking samples out of 454.

As we can see in the pictures of the Gaussians, the EM inferred a great peak for the inactive activity, in contrast with the low probability assigned to the running state. On the other hand, the GA creates a small probability for the inactivity. In fact, the first mode of the Gaussian for the inactive state is centred in (0.5322, 0.2755) with a small weight of 0.5876. The second mode is centred in (2.3928, 0.8261) and a weight of 2.11. Nevertheless, the GA creates a high probability distribution for the walking state with two Gaussian centred in (2.0002, 2.2480) and (2.7381, 1.0947) and weight of 3 and 4 respectively.

Moreover, the result of the classification by using EM is greatly dependant on the initialisation of the parameters in contrast to the GA that give stable results independently of the initial values. This is particularly useful if we do not know how the data are distributed. Furthermore, the GA counts with the powerful tool of mutation that assures that the individuals look for a wide range of solutions whereas the EM algorithm converges to a local optimum without exploring other possible far solutions.

As a conclusion, we can derive that for this case the GA outperforms the traditional EM at the expense of ignoring the running activity. Thus, for real applications, a trade-off must be done according to the results that we are interested in obtaining.

The future work aims to look for new probability distributions that defined in a better way the data and classified more properly, especially for the distinction between the walking and running classes.

References

1. Moeslund, T.B. and Granum, E. 'A Survey of Computer Vision-Based Human Motion Capture'. Computer Vision and Image Understanding, Volume 81(3), 2001, pp. 231 – 268.
2. Pavlidis, I., Morellas, V., Tsiamyrtzis, P., and Harp, S. 'Urban surveillance systems: from the laboratory to the commercial world', Proc. IEEE, 2001, 89, (10), pp. 1478–1495.
3. Rath, T.M., and Manmatha, R.: 'Features for word spotting in historical manuscripts'. Proc. of the 7th Int. Conf. on Document Analysis and Recognition, 2003, pp. 512–527.
4. Oates, T., Schmill, M.D., and Cohen, P.R.: 'A method for clustering the experiences of a mobile robot with human judgements'. Proc. of the 17th National Conf. on Artificial Intelligence and Twelfth Conf. on Innovative Applications of Artificial Intelligence, AAAI Press, 2000, pp. 846–851.
5. Yamato, J., Ohya, J., and Ishii, K., 'Recognizing human action in time-sequential images using hidden Markov model,' Proceedings of CVPR '92, pp. 379-385.
6. Wilson, A.D. and Bobick, A.F., 'Parametric hidden Markov models for gesture recognition', IEEE Transactions on Pattern Analysis and Machine Intelligence, 1999, 21, (9), pp. 884 – 900.
7. Bicego, M. and Murino, V., 'Investigating hidden Markov models' capabilities in 2D shape classification,' IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26, (2), pp. 281 – 286.
8. Nguyen, N.T., Bui, H.H., Venkatesh, S., and West, G.: 'Recognising and monitoring high-level behaviour in complex spatial environments'. IEEE Int. Conf. on Computer Vision and Pattern Recognition, Wisconsin, 2003, pp. 1–6.
9. Ivanov, Y., and Bobick, A. 'Recognition of visual activities and interaction by stochastic parsing', IEEE Trans. Pattern Recognit. Mach. Intell., 2000, Vol. 22(8), pp. 852–872.
10. Rota, N. and Thonnat, M. 'Activity Recognition from Video Sequences using Declarative Models', Proc. European Conf. on A.I., 2002, pp. 673 – 680.
11. Stamp M 'A Revealing Introduction to Hidden Markov Models' January 18, 2004, http://www.cs.sjsu.edu/faculty/stamp/RUA/HMM.pdf.
12. Bilmes, J. 1998. A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and Hidden Markov Models. Tech. Rep. ICSI-TR-97-021, University of California Berkeley.
13. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/caviar.htm
14. http://www-prima.inrialpes.fr/PETS04/pets04.html
15. http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html
16. http://www.shef.ac.uk/acse/research/ecrg/gat.html