Hierarchical unsupervised learning of facial expression categories

Jesse Hoey
Department of Computer Science
University of British Columbia
Vancouver, Canada, V6T 1Z4

Abstract

We consider the problem of unsupervised classification of temporal sequences of facial expressions in video. This problem arises in the design of an adaptive visual agent, which must be capable of identifying appropriate classes of visual events without supervision to effectively complete its tasks. We present a multilevel dynamic Bayesian network that learns the high-level dynamics of facial expressions simultaneously with models of the expressions themselves. We show how the parameters of the model can be learned in a scalable and efficient way. We present preliminary results using real video data and a class of simulated dynamic event models. The results show that our model correctly classifies the input data comparably to a standard event classification approach, while also learning the high-level model parameters.

1. Introduction

The goals of this paper are twofold. First, we wish to show that it is feasible to simultaneously learn classes of facial expression events from video input and the high-level temporal structure of these events in an unsupervised context. Second, we wish to show that learning the high-level temporal structures helps the unsupervised learning of the facial expression classes. Consider a student playing an educational computer game which is moderated by a socially intelligent agent which has inputs from a camera observing the student [6]. It is important for the agent to be able to gauge the emotional and intellectual state of the student in order to most effectively present information, but it must do so without supervision in order to adapt to new students. Although such an agent will gather information from many sources, we consider only the recognition of facial expression in this paper. We assume the user's face is tracked and spatially segmented, and that a feature vector is given by the affine parameters of optical flow extracted from each pair of frames. The goal of our system is to properly classify a set of facial expression events as represented by these feature vectors without supervision. This paper will show that if the visual events which are being clustered are related according to some higher-level dynamical structure, then learning this dynamical structure can help in the disambiguation of event classes at the lower level. The situation is analogous to word-level speech recognition, in which high-level (semantic) structure is used to guide low-level (syntactic, or phoneme-level) structure [19]. Hierarchical structures such as these can be extended to include sentence-level structures. In a similar way, the dynamics of events during one stage of an educational game might be different than during other stages. Therefore, even higher level dynamical structures at the game stage level can be learned.

A model which encompasses both high-level semantic structure and low-level syntactic structure must take temporal scales into account. In the case of an educational game, the events which make up a stage of the game include the various facial expressions of the user, each of which takes a few input video frames to complete. In this case, the time scales range from minutes at the stage level, through seconds at the facial expression level, to tenths of a second at the input video level.

This paper will present a hierarchical dynamic Bayesian network to model the visual events at each of a number of temporally abstract levels. A level in the network consists of a mixture of Markov chains (MMC). The mixture coefficients at each level are used as distributions over the parent level's states. The lowest level is a mixture of Gaussian distributions, which quantizes the affine flow feature vector, but does no temporal abstraction. Temporal scales are modeled by the characteristic length scales of each MMC, and can change dynamically. Information flow through the model is both *bottom-up* or *diagnostic* and *top-down* or *predictive*. The combination of both can be used to learn the the parameters of the model using the *expectation-maximization* algorithm [7].

The next section will review some previous work on the subject of classifying human action. Section 3 will describe the model in detail, including the parameter learn-

ing. Section 4 presents a simple set of experiments using a database of real facial expression sequences, and a simulated model of the temporal dynamics of user's facial expressions. The results of the experiments are described in Section 5, and show that the parameters of the complete model can be learned using expectation-maximization, and help in the classification of visual events in cases with significant high-level dynamical structure.

2. Related Work

The idea of combining low-level syntactical or statistical models of raw input data with high-level semantic models of conceptual states have been researched in general terms [8], and in the speech recognition literature [19]. However, typical speech recognition systems do not learn high-level semantics (at the word or sentence level), relying instead on prior knowledge. Here we review recent work in applying the same ideas to vision problems. Wren *et. al* [23] investigate understanding purposeful human motion using a combination of a kinematic model of human motion, a mixture of hidden Markov models and a high-level classifier, similar to the model proposed here. However, they do not model temporal dynamics at the high level, only classifying the hidden Markov models, resulting in a high-level model which closely resembles the mixture of HMMs [21].

Bobick and Ivanov [3] model primitive events using a set of hidden Markov models, and then use a stochastic context-free grammar (SCFG) to provide high-level semantic help. A SCFG enables the modeling of more complex semantic patterns of behavior than a hidden Markov model. Their system is trained in a supervised fashion and the high-level model is hand-coded, not learned. A similar approach [9] uses variable length Markov models (VLMMs) to capture long-range dependencies in behaviors. Sequences of human aerobics in video are captured as contours and quantized into a discrete set of prototypical feature vectors. These vectors are then modeled with variable lengths of memory in a hierarchical fashion. Vogler and Metaxas [22] describe a system to recognize American Sign Language (ASL) by combining context-dependent HMMs (HMMs with finite memory) with a three dimensional motion analysis system. The 3D motion analysis takes place in a separate, rule-based, processing stream, which is used to constrain the recognition process in the HMMs. Cohen, Garg and Huang [5] have proposed a multilevel HMM to recognize facial expressions. Their contribution is a method for automatically segmenting a video stream into facial expression events, assuming a neutral state between each expression. However, their model requires a supervised learning stage.

Eigen-analysis is the most well used approach for facial expression analysis [15], but one of the most discriminative systems is that of Tian *et al.* [14], which uses feature point tracking and a neural network. Optical flow fields are used in [2]. However, these systems are completely supervised, and do not deal with any high level structure. Furthermore, these methods develop representations specific to facial expressions, and could not be extended to include other types of human motion, which may be useful for the application we are considering.

The structure of our proposed model is closely related to that of the switching state-space model [10], in which the parameters of a linear dynamic system (LDS) are conditioned on a (discrete) state variable which lives in a Markov chain or in the output distributions of a hidden Markov model. This type of model has recently been applied to vision problems [17]. These models are similar to a two-level version of the model we propose. There are three main differences. First, these models use a continuous state space at the lower level, and discrete (switching) states and the higher level, whereas our model has discrete states at both levels. This is a modeling issue however, as continuous and discrete states can be interchanged at the lowest level in either model dependent on the task. Second, the transitions occur asynchronously at the two levels in our model. This is a reasonable assumption (the higher level states actually operate at slower time scales) which also improves efficiency, but necessitates additional constraints to specify the difference in time scales. Temporal segmentation implements these constraints in our model. Similar asynchronous models constrain the timing directly [11], or constrain the emission of observations from states [1]. Third, whereas the low-level states in the models of [10, 17], form one long continuous Markov chain, those in our model are broken into temporal sequences. The assumption is validated if one considers that the observed sequence may be temporally sampled, and so the low-level Markov chain is, indeed, broken at temporal segmentation points.

Smyth [21] and Li [13] both propose methods for clustering temporal sequences using hidden Markov models. They do not attempt to find high-level dynamical structure, only inferring a single mixture of hidden Markov models (representing a set of events). They do, however, examine the structure learning problem as well, which we do not attempt to do here, but which is planned for future stages of our research. The method of [21] is a special case of the model presented above (as is any hidden Markov model), and we compare our method to it in order to analyze the benefits of modeling high-level semantic structure.

3. Method

We use the standard notation, in which variables are denoted with uppercase letters (e.g., X, Y, Z), while particular values of variables are the same letter, but in lowercase (e.g. x, y, z). Sets or sequences of variables are written in bold-

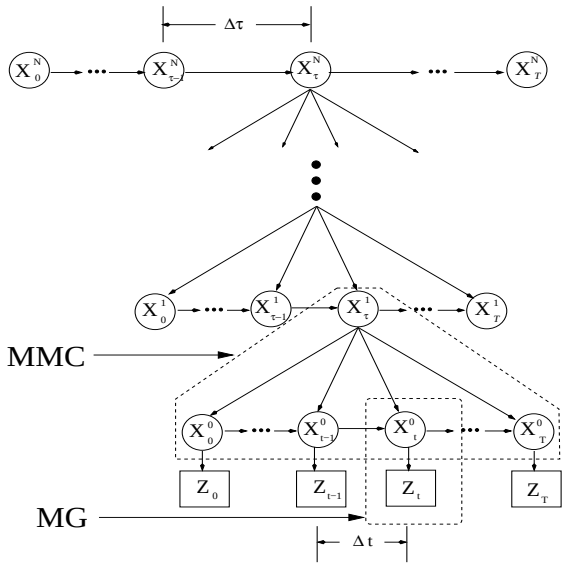

Figure 1. Hierarchical dynamic Bayesian network for unsupervised facial expression recognition.

face ($\mathbf{X} = X_0 ... X_\mathcal{T}$), while the corresponding bold-face lower-case letter ($\mathbf{x} = x_0 ... x_\mathcal{T}$) denotes an assignment of value to each variable in the set.

3.1. Model

Figure 1 shows a time slice of the proposed model as a Bayesian network. Variables X (internal node) and Z (input node) are subscripted by temporal indices and superscripted by level indices. Thus X_t^n indicates a variable at level n at time index t. As shown in the figure, the time scale at the lower level (Δt) is not the same as at the higher level ($\Delta\tau$). Two levels, n and m, with times scales $\Delta\tau$ and Δt, respectively, will have $\Delta\tau \geq \Delta t$ if $n > m$. Figure 1 shows the mixture of Gaussians (MG) at the lowest level explaining an input feature vector, Z_t, from the sequence of inputs $\mathbf{Z} = Z_0 ... Z_T$, with a distribution over discrete states at the 0-level, X_t^0. The result is a sequence of discrete states, $\mathbf{X} = X_0^0 ... X_T^0$. The time intervals at the lowest level, Δt, correspond to the period at which video frames are recorded. Figure 1 shows the fundamental level in the hierarchical model in which a class variable X_τ^1 describes the discrete sequence $\mathbf{X^0} = X_0^0 ... X_T^0$. This basic structure is a mixture of Markov chains (MMC), and we refer to the complete model as a hierarchical MMC (HMMC). The idea is that there is a sequence of T *events* at the n^{th} level, described by the variables $\mathbf{X^n} = X_0^n ... X_T^n$. The entire sequence of *events* at this level is classified as one of

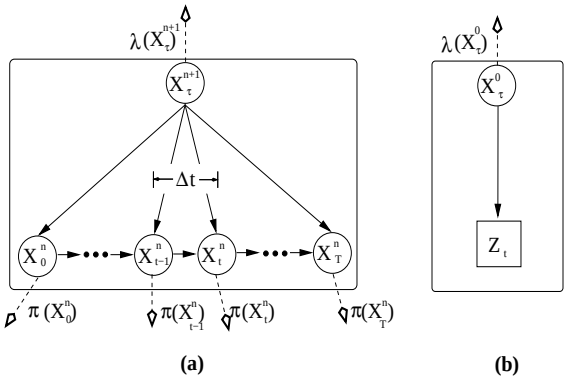

Figure 2. Mixture of Markov chains over variables X^{n+1} and X^n. The solid arrows express conditional dependence, and the dashed arrows show the message passing. (a) MMC for non-terminal levels. (b) MG for input levels.

a number of *classes*, X_τ^{n+1}. These higher-level classes are the *events* at the next higher level, occurring at (event) time τ in the $n+1^{st}$ level. The highest (N^{th}) level in Figure 1 can be considered a single-class mixture of Markov chains as well, or simply a Markov chain. A MMC in the hierarchy is thus charged with explaining *events* at its level. The model must therefore be capable of temporally segmenting the input. This will be discussed in Section 3.4.

3.2. Hierarchical Decomposition

Learning and inference in the complete Bayesian network would not be easily parallelized, and would not admit scaling to much more complex hierarchies. That is, network propagation would not be simple given the presence of many loops in the undirected graph. A common technique for dealing with such loops is to *cluster* the nodes in the graph to form compound variables in such a way that the resulting network of clusters is singly connected [18]. The general method involves forming a *join tree* over the cliques of the triangulated graph. This representation does not have a simple interpretation. However, since there are efficient computations for MMCs, the network can be clustered as shown by the dotted lines in Figure 1. Figure 2(a) shows one of the nodes of the clustered network, a mixture of Markov chains. The MMC is completely described by two parameters.

1. **Initialization** $\Theta_{0ik}^n \equiv P(X_0^n = x_i^n | X_\tau^{n+1} = x_k^{n+1})$ is the probability that $X_0^n = x_i^n$ at time 0 of the τ^{th} $n+1$-level sequence, given that $X_\tau^{n+1} = x_k^{n+1}$. At the lowest level, this becomes $\Theta_k^0 \equiv P(Z_t | X_t^0 = x_k^0)$ which is parametrized by a Gaussian distribution.

2. **Transition** $\Theta^m_{Xijk} \equiv P(X^n_t = x^n_i | X^n_{t-1} = x^n_j, X^{n+1}_\tau = x^{n+1}_k)$ describes the probability that the state $X^n_t = x^n_i$ at time t of the τ^{th} sequence given the state $X^n_{t-1} = x^n_j$ at the previous time step and the class variable $X^{n+1}_\tau = x^{n+1}_k$ at the next highest level. At the lowest level (MG), this parameter is non-existent since all sequences are of length one.

The parameters for the n^{th} level will be denoted $\Theta^n = \{\Theta^n_X, \Theta^n_0\}$ The complete set of parameters for a given model with N levels will be referred to as $\Theta = \{\Theta^0, \Theta^1, ..., \Theta^N\}$.

The MMC unit shown in Figure 2(a) as part of a complete model, as in Figure 1, must communicate with its parents and children. This is accomplished by message passing in the clustered network, which we now describe. The inputs to the input (MG) level, as shown in Figure 2(b), are sequences of continuous feature vectors, $\mathbf{z}$. The mixture of Gaussians (MG) calculates the probability of each input vector, z_t, given the Gaussian generated by the discrete state, x^0_τ. The resulting distribution passed up to the next (0^{th}) level is simply $\lambda(x^0_\tau) = P(z_t|x^0_\tau)$. All other levels are like the MMC level shown in Figure 2(a). The n^{th} level MMC receives a sequence of λ messages from the $n-1^{st}$ level below it. The messages are input distributions over the states of X^n. The MMC level will have a method for deciding when to segment the input sequence, discussed in Section 3.4. After a decision to segment, the n^{th} level computes an upward message, $\lambda(x^{n+1}_\tau) = P(e^-|x^{n+1}_\tau)$, a distribution over the states of X^{n+1} at time τ, where e^- is the *evidence* (input data) being explained by this temporal sequence. This upwards message can be expanded by summing over the possible values of $\mathbf{X}^n$ as

$$\lambda(x^{n+1}_\tau) = P(e^-|x^{n+1}_\tau) = \sum_{\mathbf{x^n}} P(e^-|\mathbf{x^n}) P(\mathbf{x^n}|x^{n+1}_\tau),$$

the terms of which can be expanded as $P(e^-|\mathbf{x^n}) = P(e^-|x^n_0 ... x^n_T) = \prod_t P(e^-_t|x^n_t) = \prod_t \lambda(x^n_t)$ and $P(\mathbf{x^n}|x^{n+1}_\tau) = \prod_t P(x^n_t|x^n_{t-1}, x^{n+1}_\tau) P(x^n_0|x^{n+1}_\tau)$, giving the upwards message from the level shown in Figure 2(a) as

$$\lambda(x^{n+1}_\tau) = \sum_{\mathbf{x^n}} \prod_t \lambda(x^n_t) \Theta^n_{X x^n_t x^n_{t-1} x^{n+1}_\tau} \Theta^n_{0 x^n_0 x^{n+1}_\tau}. \quad (1)$$

This upwards going message is complemented by a downwards going message from the $n+1^{st}$ level, which is either from a higher level, or from a prior distribution at the highest level. The downwards message initiates the computation of a predicted distribution over the states of the variable X^n at the next time step, $\pi(x^n_t) = P(\mathbf{x^n}|e^+)$, where the *evidence*, e^+, in this term comes from two places: $e^+_<$ and $e^+_>$, which are all the input data subsumed by the states x^n_τ for $t < \tau$ and $t > \tau$, respectively. There is a third component of *evidence*, e^+, which comes from the ancestors of the $n+1^{st}$ level. Thus, the downwards message can be factored as

$$\begin{aligned} \pi(x^n_\tau) &= P(x^n_\tau | e^+_< e^+_> e^+) \\ &= P(e^+_> | x^n_\tau e^+_< e^+) P(x^n_\tau | e^+_< e^+) \\ &= P(e^+_> | x^n_\tau e^+) P(x^n_\tau | e^+_< e^+) \end{aligned} \quad (2)$$

Equation (2) is a product of two terms. The first is a *backward* variable, which gives the probability of all future data given the present state, while the second is a *forward* variable, giving the probability of the present state given all past data. These terms are analogous to the familiar forward and backward terms in the Baum-Welsh training procedure from the HMM literature [19]. Equation (2) poses a dilemma. While the both terms in (2) can be efficiently computed, only the second can be done at time τ. This will cause some difficulties in the parameter learning algorithm, and will be discussed in Section 3.3.

Upwards and downwards messages can be combined at each level to calculate *belief states* for each set of variables in the hierarchy. These belief states will be used for learning the parameter values from data. The belief state over a particular variable, say x^n_τ, given all evidence from below (the observations) and from above (prior information) is computed as $BEL(x^n_\tau) = P(x^n_\tau|e^+, e^-) = \alpha P(e^-|x^n_\tau, e^+) P(x^n_\tau|e^+)$ where α is a constant of proportionality. Using the notation of (2) again, this is

$$P(x^n_\tau | e^+_< e^+_> e^+ e^-) = P(e^-|x^n_\tau) P(x^n_\tau | e^+_< e^+_> e+) = \lambda(x^n_\tau) \pi(x^n_\tau) \quad (3)$$

The upwards flow of information classifies the input at each level in the hierarchy. That is, a given input sequence has some maximum likelihood state at each level. The downwards flow of information gives a probability distribution (prediction) over the next time steps at each level.

3.3. Learning the parameters

The last section showed how information was propagated through the HMMC presented in Figure 1. This section shows how this information can be used to learn the parameters of the model given input data. We present the learning algorithm as an application of the *expectation-maximization* (EM) algorithm [7]. We show how the EM algorithm naturally factors according to our model. We then discuss an online version, which does not pass downwards information from backward passes at all levels.

Given a set of observation sequences, $\mathbf{Z}$, we wish to find the values of Θ which maximize

$$P(\mathbf{Z}\mathcal{E}\Theta) = \int_{\mathbf{X}} P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)$$

where the integral is over all the *hidden* variables in an N level model, $\mathbf{X} = \{X^0, ..., X^N\}$. Any further evidence from above the top level (*e.g* user controls) are denoted $\mathcal{E}$. The optimization can be performed using the expectation-maximization (EM) algorithm [7].

The function $f(\Theta) = P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)$ is lower bounded with a function $q(\mathbf{X})$:

$$f(\Theta) = \int_{\mathbf{X}} \frac{P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)}{q(\mathbf{X})} q(\mathbf{X}) \geq \prod_{\mathbf{X}} \left(\frac{P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)}{q(\mathbf{X})} \right)^{q(\mathbf{X})}, \tag{4}$$

where the inequality is given by Jensen's inequality. Taking logarithms of the right side we get

$$G(\Theta, q) = \int_{\mathbf{X}} q(\mathbf{X}) \log P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta) - q(\mathbf{X}) \log q(\mathbf{X}). \tag{5}$$

Then, at the current guess for Θ, Θ', we choose q to maximize G, so that G touches f at Θ'. We use the constraint that $\int_{\mathbf{X}} q(\mathbf{X}) = 1$ and get

$$q(\mathbf{X}) = \frac{P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)}{\int_{\mathbf{X}} P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)} = \frac{P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)}{P(\mathbf{Z}\mathcal{E}\Theta)} = P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta)$$

Calculating $P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta)$ is the "E" step of EM. This quantity can be computed as $P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta) = P(\mathbf{X^0}|\mathbf{X^1}\mathbf{Z}\mathcal{E}\Theta)P(\mathbf{X^1}|\mathbf{X^2}\mathbf{Z}\mathcal{E}\Theta)...P(\mathbf{X^N}|\mathbf{Z}\mathcal{E}\Theta)$, and is the product of the belief states at each level, which we know can be computed from upwards and downwards messages, However, they need not be computed explicitly, as we will show in the next section.

After computing $q(\mathbf{X}) = P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta)$, we can update the values of the parameters by maximizing (5):

$$\int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta') \log P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta). \tag{6}$$

with respect to Θ. This is called the "M" step of the EM algorithm.

According to the independence relationships shown in the Bayesian network in Figure 1, we can factor $P(\mathbf{Z}\mathbf{X}\mathcal{E}\Theta)$ as $P(\mathbf{Z}\mathbf{X^0}\Theta^0|\mathbf{X^1}\Theta^1)P(\mathbf{X^1}\Theta^1|\mathbf{X^2}\Theta^2)...P(\mathbf{X^N}\mathcal{E}\Theta^N)$, so that, writing the log product as a sum of logs, (6) becomes:

$$\begin{aligned}
& \int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta) \log P(\mathbf{Z}\mathbf{X^0}\Theta^0|\mathbf{X^1}\Theta^1) \\
+ & \int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta) \log P(\mathbf{X^1}\Theta^1|\mathbf{X^2}\Theta^2) \\
+ & \vdots \\
+ & \int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta) \log P(\mathbf{X^n}\Theta^n|\mathbf{X^{n+1}}\Theta^{n+1}) \\
+ & \vdots \\
+ & \int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Z}\mathcal{E}\Theta) \log P(\mathbf{X^N}\mathcal{E}\Theta^N)
\end{aligned} \tag{7}$$

The sums over $\mathbf{X^0}...\mathbf{X^N} - \mathbf{1}$ can be performed in the last term, leaving

$$\int_{\mathbf{X^N}} P(\mathbf{X^N}|\mathbf{Z}\mathcal{E}\Theta') \log P(\mathbf{X^N}\mathbf{Z}\mathcal{E}\Theta^N) \tag{8}$$

This term can be maximized in the N^{th} level, and constitutes the "M" step for that level by itself. The sums over $\mathbf{X^N}...\mathbf{X^{n+2}}, \mathbf{X^{n-1}}...\mathbf{X^0}$ can be performed in the n^{th} term in (7), which can then be written as:

$$\int_{\mathbf{X^n}\mathbf{X^{n+1}}} P(\mathbf{X^n}\mathbf{X^{n+1}}|\mathbf{Z}\mathcal{E}\Theta') log P(\mathbf{X^n}\mathbf{Z}|\Theta^n\mathbf{X^{n+1}}\Theta^{n+1}) P(\Theta^n). \tag{9}$$

Maximizing each such term in (7) is the "M" step of the EM algorithm for each level by itself. The following section describes how to perform this maximization in the case of multinomial distributions.

Our goal now is to maximize all equations like (9) (of which (8) can be considered a special case) with respect to the parameters $\Theta^0...\Theta^N$ We assume the likelihood function of the completed data is given by a multinomial distribution:

$$P(\mathbf{Z}\mathbf{X}|\Theta) = \prod_n \prod_{ijk} \Theta_{Xijk}^{n\,M^n_{Xijk}} \Theta_{0ik}^{n\,M^n_{0ik}},$$

where the Ms are the *sufficient statistics* for the multinomial distributions, *e.g.* M^n_{Xijk} is the number of times $X^n_t = i$ when $X^n_{t-1} = j$ and $X^{n+1} = k$ in the data. Then, we can set the derivative of (9) with respect to a particular Θ^n_{Xijk} to 0, subject to the constraint that $\sum_i \Theta^n_{Xijk} = 1$, and find that

$$\Theta^n_{Xijk} = \frac{\alpha^n_{Xijk} + E_{P(\mathbf{X^n}\mathbf{X^{n+1}}|\mathbf{Z},\mathcal{E}\Theta')}(M^n_{Xijk})}{\sum_i \alpha^n_{Xijk} + E_{P(\mathbf{X^n}\mathbf{X^{n+1}}|\mathbf{Z},\mathcal{E}\Theta')}(M^n_{Xijk})} \tag{10}$$

where α^n_X is the multinomial conjugate (Dirichlet) prior for the parameter Θ^n_X The expectation in (10) is

$$E_{P(\mathbf{X^n}\mathbf{X^{n+1}}|\mathbf{Z}\Theta')}(M^n_{Xijk}) = \sum_{\tau=1}^{\mathcal{T}} P(X^{n+1}_{\tau,k}|\mathbf{Z}\mathcal{E}\Theta') \sum_{t=1}^{T} P(X^n_{t,i}, X^n_{t-1,j}|X^{n+1}_{\tau,k}\mathbf{Z}\Theta^n)$$

The sums are over the time indices, t and τ, at the n^{th} and $n+1^{st}$ level, respectively. The expectation is therefore a sum of all the probabilities of seeing $X^n_{t,i}$ and $X^n_{t-1,j}$ for some t at level n, given that $X^{n+1}_\tau = k$, weighted by the probability of seeing $X^{n+1}_\tau = k$. While the sums over t at the n^{th} level can be performed efficiently, given the value of X^{n+1}_τ, by using a forward-backward decomposition, the weighting terms, $P(X^{n+1}_{\tau,k}|\mathbf{Z}\mathcal{E}\Theta')$ depend on data not yet seen at time τ at the $n+1^{st}$ level. Indeed, these weighting terms are no more than the *belief states* we have already seen in (3). As previously noted, the *backwards* part of the

downwards message (2) cannot be calculated exactly at time τ. Thus, the entire model could not be updated until all data had been seen, at which point the full EM algorithm could be run to completion. Clearly this is infeasible, as the highest level time scales may be extremely long in general.

There are a number of options for dealing with this problem. First and simplest, we can just forget about it, and use only forward information in the downwards messages. Second, each level can compute downwards messages only after a complete sequence has been observed (at the given level) and the backwards pass has been computed, but before the parent level has passed down information from its backwards pass (hence only using predictions from the parent level's forward pass). This option is used in our current implementation. This idea can be extended so that a level will wait for a more remote ancestor level to complete its backwards pass before computing downwards messages. In fact, the ancestor level may not need to compute a backwards pass over a complete sequence, but only over some relevantly recent portion of it, as indicated by some discount factor (*online EM* [4]). A further extension could use information from ancestor level backwards passes from previous EM iterations (*generalized EM* [16]).

3.4. Temporal Segmentation

The model we have described relies on a segmentation at each level in the hierarchy. The dynamic time-warping capabilities of HMMs means that segmentation is not restricted to a particular time interval at each level, and can be dynamically specified as a sequence is being analyzed. We are currently using a manual segmentation into sequences which are representative of the user's current affective or emotional state. In practice, such a segmentation could be performed automatically using, for example, the methods of Rui and Anandan [20]. The temporal segmentation could also be incorporated into the learning process, as in [9, 22]. Finally, the temporal segmentation could be achieved by sampling the input data stream. It is important to note that although temporal segmentation could be achieved automatically at the lowest level, where the observation vectors have a precise meaning in terms of the observed sequence, the segmentation at higher levels is much less well defined.

4. Experiments

This section will describe preliminary experiments designed to test our hypotheses that the hierarchical mixture of Markov chains (HMMC) learns to recognize facial expression events and the high-level temporal dynamics of these events, and that the additional information learned by the high-level model increases the recognition rates. We use a simulated model of the temporal progression of facial

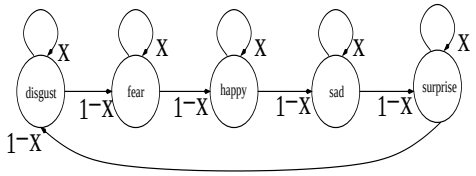

Figure 4. Simulation model. States correspond to facial expressions. Arrows denote probabilistic transitions with probabilities x.

expressions, as shown in Figure 4. The facial expressions of the user follow one another in a predetermined order (a left-right model with return), with transitions between states governed by a parameter $x \in [0, 0.5]$. The simulated dynamics shown in Figure 4 were designed so that the parameter x governs how much structure there is at the high level. Thus, $x = 0$ is a very structured model (transitions are deterministic), while $x = 0.5$ is a model with less structure [1]. A state in the model generates a video sequence of a subject performing the corresponding facial expression. It is important to note that the model is not meant as a model of human emotional progression, but only as a simulator in which a parameter (x) governs the predictability of the sequence of events in some way. The goal was to simulate a well structured sequence of fairly distinguishable facial expression sequences, in order to assess what effects the learning of high level structure has on the unsupervised recognition problem.

The video sequences are taken from a database of 69 sequences of a single subject performing the 5 expressions. They are processed by first extracting optical flow and then projecting onto the affine basis. The projections are computed as a simple dot product over the face area, normalized by area. The expressions were performed by a single, untrained, subject who was simulating the 5 emotional states. Figure 3 shows selected frames from two sequences. It is the same database as used in a supervised version of the facial expression recognition problem [12].

We learn two models from a given set of input data: a two level version of the HMMC model presented in Section 3, and a mixture of hidden Markov models as described in [21]. There are three main differences between the two models. The first is that the HMMC can take advantage of any temporal structure at the high level. The second is that mixture of HMMs has a separate set of Gaussian output distributions for each combination of high and low level states, whereas the high-level states in our model share Gaussian distributions (known as *parameter tying*, re-

[1] Models in this class are more difficult to simulate for $x \in [0.5, 1]$, as some states may never be reached, causing initialization problems.

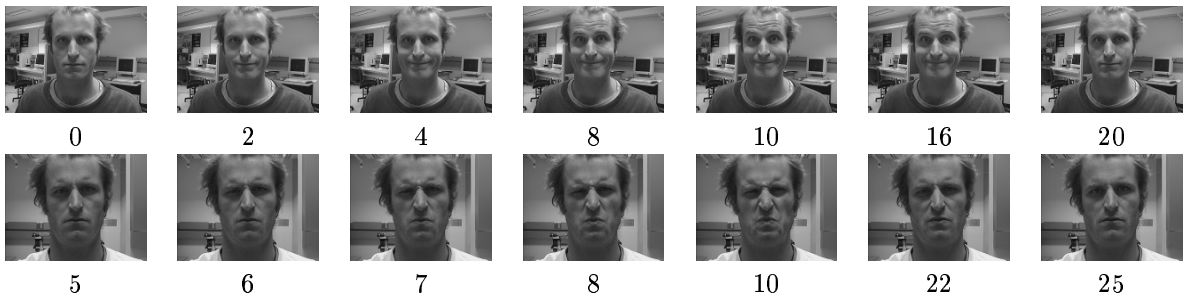

Figure 3. Selected frames from a *happy* (top) and a *disgust* (bottom) expression sequence

sulting in *semi-continuous* models). The third difference is that the mixture of HMMs is performing complete expectation steps, whereas our model only does partial expectation steps. While the first difference is expected to increase the effectiveness of our model, the second two are expected to decrease it. We will show that the benefits outweigh the disadvantages for well structured models ($x \to 0$ in Figure 4)

The model described in Section 3 has been implemented in Java. An abstract `Level` class is at the top of the class hierarchy, and is sub-classed by `MG` and `MMC` classes. Each level runs its in its own thread of execution, making the application distributable and scalable. The MG level in the HMMC was initialized by performing vector quantization on the input data space. The lowest MMC level in the HMMC was initialized using a heuristic technique which makes histograms of the values of X and runs K-means on the resulting vectors. The top level MMC was randomly initialized. The mixture of HMMs model was initialized with the same Gaussian distributions as in the MG level (replicated) and with the same initialization and transition parameters in the lowest MMC level. The initialization method of [21], which involves clustering the data in log likelihood space first, may give better results, but we assume it will benefit both models equally.

We performed 200 trials with different initial random seeds for the initialization procedure. Each trial uses a leave-one-out procedure. The training data consists of all sequences but one (chosen at random) from each motion class. The model in Figure 4 is simulated for about 250 steps during which video sequences are chosen sequentially (with return) from the training set. Expectation-maximization is performed using the same data until convergence is reached. The trained models are then tested by again simulating for about 250 steps. This time, however, the outputs are chosen from the 4 sequences which were left out of the training data. Thus, the same sequence is presented to the system when the simulator is in each of the states in Figure 4. The most likely state sequence over the top level is determined using the Viterbi algorithm, and compared with the (known) sequence of states in the simulated model. Classification accuracy in an unsupervised context is not simple to define. However, the trained model only needs a one-to-one correspondence between its classified states and the actual environmental states, but the order of the correspondence does not matter. We determine the ordering which gives the best recognition rate after the training phase, and use this ordering to measure classification accuracy of training and of test phases.

5. Results

Figure 5 shows typical log-likelihoods as a function of EM iteration during learning of the mixture of hidden Markov models and for the HMMC. The mixture of HMMs outperforms the HMMC in modeling the training data faithfully. This is primarily due to the sub-optimality of the expectation step as described above. [2]

Table 1 shows classification accuracies for training and test sequences using both models under consideration. All percentages are averages over the 200 trials, each of which has randomized initialization parameters and a random set of left out sequences. High values of the transition parameter, x, are models with more structure, which the HMMC can take advantage of, but the mixture of hidden Markov models cannot. We see that, as expected, our hierarchical model outperforms the mixture of HMMs model for low values of $x = 0.0 - 0.3$. Thus, the high-level semantic model, when properly learned, disambiguates the facial expression input sequences. The HMMC performs more poorly for intermediate values of $x = 0.4 - 0.5$. The simulated model in these cases has less structure, and our learning procedure is at a disadvantage because it is only performing an approximate expectation step. We expect that the full EM procedure would give better results for any value of x, but would require additional resources and pose time constraints which are not desirable for our application.

To evaluate how well the unsupervised learning was performing, we performed a supervised experiment. The ex-

[2] The differences in log likelihood in the trained model seem large (10^4), but the sequences being modeled consist of about 250 facial expression sequences, each about 20-30 frames in length. Therefore, the log likelihood difference per sequence is on the order of 100.

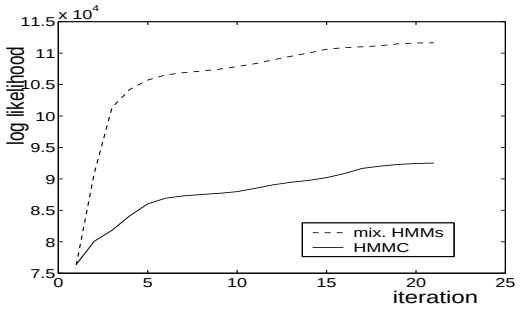

Figure 5. Typical log likelihood evolution curves for learning the model in Figure 4 ($x = 0$) using mixture of HMMS and HMMC.

x	HMMC		Mixture of HMM	
	train	test	train	test
0.0	92	91	81	81
0.1	91	89	81	82
0.2	91	88	81	84
0.3	88	84	80	82
0.4	88	83	81	82
0.5	87	81	82	83

Table 1. Classification accuracies (%) for training and test sequences using the HMMC and the mixture of HMMs presented in [21].

pression sequences were labeled and used to train a set of 5 HMMs, one for each expression. The initialization procedure was identical. Recognition was performed by choosing the model which gave the highest likelihood of the observations. A leave-one-out procedure was also used for this experiment, and the average recognition rate was 98%. Unsupervised learning decreases recognition rates, because the learning procedure must uncover both the categories of expression *and* learn the models within each category. However, if we also model the high-level dynamics in a temporally well structured environment, we can achieve recognition rates approaching the supervised case (to within 7%). Without these high-level dynamics, the recognition rates drop (to below 14% less than the supervised case).

6. Conclusions

We presented a hierarchical dynamic Bayesian network for unsupervised classification of expression sequences. We showed how to simultaneously learn low and high level models of events. Current work is focussing on automatic segmentation, on recognizing subtle facial expressions, and on using expressions to guide a tutoring agent.

References

[1] S. Bengio and Y. Bengio. An EM algorithm for asynchronous input/output hidden Markov models. In *Int. Conf. Neural Information Processing Systems*, Hong Kong, 1996.

[2] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motions. *IJCV*, 25(1):23–48, 1997.

[3] A. Bobick and Y. Ivanov. Action recognition using probabilistic parsing. In *IEEE ICCV*, Mumbai, India, 1998.

[4] X. Boyen and D. Koller. Approximate learning of dynamic models. In *NIPS 11*, 1998.

[5] I. Cohen, A. Garg, and T. Huang. Emotion recognition from facial expressions using multilevel HMM. In *NIPS 14*, 2000.

[6] C. Conati. Probabilistic assessment of user's emotions during the interaction with educational games. In *Proc. Workshop on Attitude, Personality and Emotions in User Adaptive Interaction*, UM 2001.

[7] A. Dempster, N.M.Laird, and D. Rubin. Maximum likelihood from incomplete data using the EM algorithm. *Journal of the Royal Statistical Society*, 39(B), 1977.

[8] K. Fu. A step towards unification of syntactic and statistical pattern recognition. *IEEE PAMI*, 8(3):396–404, May 1986.

[9] A. Galata, N. Johnson, and D. Hogg. Learning structured behaviour models using variable length Markov models. In *IEEE Workshop on Modelling People*, Corfu, Sept. 1999.

[10] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):963–996, 1998.

[11] S. E. Hihi and Y. Bengio. Hierarchical recurrent neural networks for long-term dependencies. In *NIPS 8*, 1996.

[12] J. Hoey and J. J. Little. Representation and recognition of complex human motion. In *IEEE CVPR*, June 2000.

[13] C. Li and G. Biswas. A Bayesian approach to temporal data clustering using hidden Markov models. In *ICML 2000*

[14] Y. Tian, T. Kanade, and J. Cohn. Recognizing action units for facial expression analysis. *IEEE PAMI*, 23(2), Feb. 2001.

[15] M.Turk and A.Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[16] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental and other variants. Technical report, Dept. of Comp. Sci., Univ. of Toronto, 1993.

[17] V. Pavlovic, B. J. Frey, and T. S. Huang. Variational learning in mixed-state dynamic graphical models. In *Proc. UAI 99*

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[19] L. Rabiner and B. Huang. *Fundamentals of Speech Recognition.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[20] Y. Rui and P. Anandan. Segmenting visual actions based on spatio-temporal motion patterns. In *IEEE CVPR*, June 2000.

[21] P. Smyth. Clustering sequences with hidden Markov models. In *NIPS 10*, 1997.

[22] C. Vogler and D. Metaxas. ASL recognition based on a coupling between HMMs and 3D motion analysis. In *IEEE ICCV*, Mumbai, India, 1998.

[23] C. R. Wren, B. P. Clarkson, and A. P. Pentland. Understanding purposeful human motion. In *Proc. Face and Gesture Recognition*, Grenoble, France, 2000.