Nonlinear PHMMs for the Interpretation of Parameterized Gesture

Andrew D. Wilson Aaron F. Bobick
Vision and Modeling Group
MIT Media Laboratory
20 Ames St., Cambridge, MA 02139
(drew, bobick@media.mit.edu)

Abstract

In previous work [14], we modify the hidden Markov model (HMM) framework to incorporate a global parametric variation in the output probabilities of the states of the HMM. Development of the parametric hidden Markov model (PHMM) was motivated by the task of simultaneously recognizing and interpreting gestures that exhibit meaningful variation. With standard HMMs, such global variation confounds the recognition process. The original PHMM approach assumes a linear dependence of output density means on the global parameter. In this paper we extend the PHMM to handle arbitrary smooth (nonlinear) dependencies. We show a generalized expectation-maximization (GEM) algorithm for training the PHMM and a GEM algorithm to simultaneously recognize the gesture and estimate the value of the parameter. We present results on a pointing gesture, where the nonlinear approach permits the natural azimuth/elevation parameterization of pointing direction.

1 Introduction

In [14] we introduce parametric hidden Markov models (PHMMs) as a technique to simultaneously recognize and interpret *parameterized gesture*. By parameterized gesture we mean gestures that exhibit a meaningful variation; an example is a point gesture where the important parameter is direction. A point gesture is then parameterized by two values: the Cartesian coordinates that indicate direction. Alternatively, direction can be specified by spherical coordinates.

We refer the reader to [14] for a detailed motivation of the PHMM approach as it relates to gesture recognition and interpretation. We briefly mention here that without resorting to manual tinkering with the feature space, a standard dynamic time warping (DTW) or HMM approach to the recognition of parameterized gestures faces the difficulty that either the variability controlled by the parameter must be modeled as noise, or there must be an independent recognition procedure for each significantly distinct parameter value. Likewise the process of recovering the parameters necessarily involves recognizing the gesture.

PHMMs extend the standard HMM model to include a global parametric variation in the output of the HMM states. In [14] a linear model was used to model the parametric variation of the output densities at each state of the HMM. Using the linear model, we formulated an expectation-maximization (EM) method for training the parametric HMM. During testing, the PHMM simultaneously recognizes the gesture and estimates the quantifying parameters, also by an EM procedure.

After reviewing linear PHMMs, we present the extension of the framework to handle situations in which the dependence of the state output distributions on the parameters is not linear. Nonlinear PHMMs model the dependence using a single 3-layer logistic neural network at each state; this model reduces the constraint on the mapping from parameterization to output densities from being linear to simply being smooth. The nonlinear PHMM is thus able to model a larger class of gesture and movement than the linear PHMM, and by the same token, the parameterization may be chosen more freely in relation to the observation feature space.

2 Related work

Hidden Markov models and related statistical time-warping techniques have been applied to the problem of gesture recognition with notable success [11, 12, 5, 15]. None of these works has developed representations to learn meaningful variation of the gestures. For example, Starner and Pentland restrict the ASL alphabet to repeatable, non-varying gestures. In fact ASL is subject to complex grammatical processes that operate on multiple simultaneous levels. These kinds of variation in ASL are addressed in a machine perception framework by Poizner et al. [9].

In [13], we apply HMMs to the task of hand gesture recognition from video by training an eigenvector basis set of the images corresponding to each state. An image's membership to each state is a function of the residual of the reconstruction of the image using the state's eigenvectors. The state membership is thus invariant to variance along the eigenvectors. Although not applied to images directly, the present work is an extension of this earlier work in that the goal is to recover a parameterization of the systematic variation of the gesture.

Murase and Nayar [7] parameterize meaningful variation in the appearance of images by computing a representation of the nonlinear manifold of the images in an eigenspace of the images. Their work is similar to ours in that training assumes that each input feature vector is labeled with the value of the parameterization. In testing, an unknown image is projected onto the manifold and the parameterization is recovered. Their framework has been used, for example, to recover the camera angle relative to a known object in the field of view.

Recently there has been interest in methods that recover latent parameterizations. In his "family discovery" paradigm, Omohundro [8], for example, outlines a variety

of approaches to learning the nonlinear manifold representing systematic variation. One of these techniques has been applied to the task of lip reading by Bregler and Omohundro [4]. Bishop, Svensen and Williams [3] have also introduced techniques to learn latent parameterizations.

Finally, a number of systems have been developed which use gesture recognition within an interactive context. These are particularly relevant to the present work in that the system is charged with the task of extracting a parameter important to the interaction as well as the task of recognizing that the gesture occurred. The Perseus [6] system is an example. The typical approach of these systems is to first identify static configurations of the user's body that are diagnostic of the gesture, and then use an unrelated method to extract the parameter of interest (e.g., direction of pointing). Manually constructed *ad hoc* procedures are typically used to identify the diagnostic configuration, a task complicated by the requirement that this procedure work through the range of meaningful variation and also not be confused by other gestures. Perseus, for example, understands pointing gestures by detecting when the user's arm is extended. The system then finds the pointing direction by computing the line from the head to the user's hand.

3 Linear PHMMs

This section reviews the linear derivation found in [14]. PHMMs model the dependence on the parameter of interest explicitly. We begin with the usual HMM formulation [10] and change the form of the output probability distribution (usually a normal distribution or a mixture model) to depend on the parameter $\boldsymbol{\theta}$, a vector quantity.

3.1 Model

In the standard continuous HMM model, a sequence is represented by movement through a set of hidden states. The Markov property is encoded in a set of transition probabilities, with $a_{ij} = P(q_t = j \mid q_{t-1} = i)$ being the probability of moving to state j at time t given the system was in state i at time $t-1$. Associated with each state j is an output distribution of the feature vector $\mathbf{x}$ given the system is really in state j at time t: $P(\mathbf{x}_t \mid q_t = j)$. In a simple Gaussian HMM, the parameters to be estimated are the a_{ij}, $\boldsymbol{\mu}_j$, and $\boldsymbol{\Sigma}_j$.[1]

To introduce the parameterization on $\boldsymbol{\theta}$ we modify the output distributions. The simplest useful model is a linear dependence of the mean of the Gaussian on $\boldsymbol{\theta}$. For each state j of the HMM we have:

$$\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}) = W_j \boldsymbol{\theta} + \bar{\boldsymbol{\mu}}_j \tag{1}$$

$$P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t, \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}), \boldsymbol{\Sigma}_j) \tag{2}$$

In the work presented here all values of $\boldsymbol{\theta}$ are considered equally likely and so the prior $P(\boldsymbol{\theta} \mid q_t = j)$ is ignored.

Note that $\boldsymbol{\theta}$ is constant for the entire observation sequence, but is free to vary from sequence to sequence. When necessary, we write the value of $\boldsymbol{\theta}$ associated with a particular sequence k as $\boldsymbol{\theta}_k$.

Figure 1 shows the PHMM architecture as a Bayes network. Bengio and Frasconi's [1] Input Output HMM (IOHMM) is a similar architecture that maps input sequences to output sequences using a recurrent neural net, which by the Markov assumption needs only consider the current and previous time steps of the input and output.The PHMM architecture differs in that it maps a single parameter value to a sequence. Thus the parameter provides a *global* constraint on the sequences, and so the PHMM testing phase must consider the entire sequence at once. Later, we show how this feature provides robustness to noise.

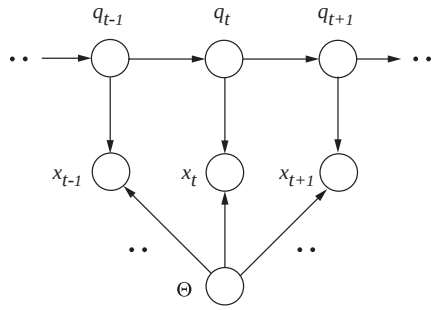

Figure 1: Bayes network showing the conditional dependencies of the PHMM.

3.2 Training

Training consists of setting the HMM parameters to maximize the probability of the training sequences. Each training sequence is paired with a value of $\boldsymbol{\theta}$. The Baum-Welch form of the expectation-maximization (EM) algorithm is used to update the parameters of the output probability distributions. The expectation step of the Baum-Welch algorithm (also known as the "forward/backward" algorithm) computes the probability that the HMM was in state j at time t given the entire sequence $\mathbf{x}_t$, denoted as γ_{tj}. It is convenient to consider the HMM's parse of the observation sequence as being represented by the matrix γ_{tj}.

In training, the parameters ϕ of the HMM are updated in the maximization step of the EM algorithm. In particular, the parameters ϕ are updated by choosing a ϕ' to maximize the auxiliary function $Q(\phi' \mid \phi)$, the expected value of the log probability of the sequence given the parse γ_{tj}. ϕ' may contain all the parameters in ϕ, or only a subset if several maximization steps are required to estimate all the parameters. In [14] we derive Q for HMMs:

$$\frac{\partial Q}{\partial \phi'} = \sum_t \sum_j \gamma_{tj} \frac{\frac{\partial}{\partial \phi'} P(\mathbf{x}_t \mid q_t = j, \phi')}{P(\mathbf{x}_t \mid q_t = j, \phi')} \tag{3}$$

The parameters ϕ of the parameterized Gaussian HMM include W_j, $\bar{\boldsymbol{\mu}}_j$, $\boldsymbol{\Sigma}_j$ and the Markov model transition probabilities. Updating W_j and $\bar{\boldsymbol{\mu}}_j$ separately has the drawback that when estimating W_j only the old value of $\bar{\boldsymbol{\mu}}_j$ is available, and similarly if $\bar{\boldsymbol{\mu}}_j$ is estimated first. Instead, we define new variables:

$$Z_j \equiv \left[W_j \; \bar{\boldsymbol{\mu}}_j \right] \quad \boldsymbol{\Omega}_k \equiv \begin{bmatrix} \boldsymbol{\theta}_k \\ 1 \end{bmatrix} \tag{4}$$

[1] Technically there are also the initial state parameters π_j to be estimated; in this work we use causal topologies with a unique starting state.

such that $\hat{\boldsymbol{\mu}}_j = Z_j \Omega_k$. We then need to only update Z_j in the maximization step for the means.

To derive an update equation for Z_j we maximize Q by setting equation 3 to zero (selecting Z_j as the parameters in ϕ') and solving for Z_j. Note that because each observation sequence k in the training set is associated with a particular $\boldsymbol{\theta}_k$, we can consider all observation sequences in the training set before updating Z_j. Accordingly we denote γ_{tj} associated with sequence k as γ_{ktj}. Substituting the Gaussian distribution and the definition of $\hat{\boldsymbol{\mu}}_j = Z_j \Omega_k$ into equation 3:

$$\frac{\partial Q}{\partial Z_j} = \sum_k \sum_t \gamma_{ktj} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k)}{\partial Z_j} \quad (5)$$

$$= \Sigma_j^{-1} \sum_k \sum_t \gamma_{ktj} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k)) \Omega_k^T \quad (6)$$

$$= \Sigma_j^{-1} \left[\sum_{k,t} \gamma_{ktj} \mathbf{x}_{kt} \Omega_k^T - \sum_{k,t} \gamma_{ktj} Z_j \Omega_k \Omega_k^T \right] \quad (7)$$

Setting this derivative to zero and solving for Z_j, we get the update equation for Z_j:

$$Z_j = \left[\sum_{k,t} \gamma_{ktj} \mathbf{x}_{kt} \Omega_k^T \right] \left[\sum_{k,t} \gamma_{ktj} \Omega_k \Omega_k^T \right]^{-1} \quad (8)$$

Once the means are estimated, the covariance matrices Σ_j are updated in the usual way:

$$\Sigma_j = \sum_{k,t} \frac{\gamma_{ktj}}{\sum_t \gamma_{ktj}} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}_k))^T \quad (9)$$

as is the matrix of transition probabilities [10].

3.3 Testing

In testing we are given an HMM and an input sequence. We wish to compute the value of $\boldsymbol{\theta}$ and the probability that the HMM produced the sequence. As compared to the usual HMM formulation, the PHMM's testing procedure is complicated by the dependence of the parse on the unknown $\boldsymbol{\theta}$. Here we present only a technique to extract the value of $\boldsymbol{\theta}$, since for a given value of $\boldsymbol{\theta}$ the probability of the sequence $\mathbf{x}_t$ is easily computed by the Viterbi algorithm or by the forward/backward algorithm.

We desire the value of $\boldsymbol{\theta}$ which maximizes the probability of the observation sequence. Again an EM algorithm is appropriate: the expectation step is the same forward/backward algorithm used in training. The forward/backward algorithm computes the optimal parse given a value of $\boldsymbol{\theta}$. In the corresponding maximization step we update $\boldsymbol{\theta}$ to maximize Q, the log probability of the sequence given the parse γ_{tj}.

To derive an update equation for $\boldsymbol{\theta}$, we start with the derivative in equation 3 from the previous section and select $\boldsymbol{\theta}$ as ϕ'. As with Z_j, only the means $\hat{\boldsymbol{\mu}}_j$ depend upon $\boldsymbol{\theta}$ yielding:

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \quad (10)$$

Setting this derivative to zero and solving for $\boldsymbol{\theta}$, we have:

$$\boldsymbol{\theta} = \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} W_j \right]^{-1} \left[\sum_{t,j} \gamma_{tj} W_j^T \Sigma_j^{-1} (\mathbf{x}_t - \bar{\boldsymbol{\mu}}_j) \right] \quad (11)$$

The values of γ_{tj} and $\boldsymbol{\theta}$ are iteratively updated until the change in $\boldsymbol{\theta}$ is small. With the examples we have tried, less than ten iterations are sufficient. Note that for efficiency, many of the inner terms of the above expression may be pre-computed.

4 Nonlinear PHMMs

4.1 Model

Nonlinear PHMMs omit the linear model of section 3.1 in favor of a logistic neural network with one hidden layer. As with linear PHMMs, the output of each state is assumed to be Gaussian:

$$P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}_t, \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}), \Sigma_j) \quad (12)$$

The mean $\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})$ is defined to be the output of the network associated with state j:

$$\hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}) = W^{(2,j)} g(W^{(1,j)} \boldsymbol{\theta} + b^{(1,j)}) + b^{(2,j)} \quad (13)$$

where $W^{(1,j)}$ denotes the matrix of weights from the input layer to the layer of hidden logistic units, $b^{(1,j)}$ the biases at each input unit, and $g(\cdot)$ the vector-valued function that computes the logistic function of each component of its argument. Similarly, $W^{(2,j)}$ and $b^{(2,j)}$ denote the weights and biases for the output layer (see [2]).

4.2 Training

As with linear PHMMs, the parameters of the nonlinear PHMM are updated in the maximization step of the training EM algorithm by choosing ϕ' to maximize the auxiliary function $Q(\phi' \mid \phi)$.

In the nonlinear PHMM, the parameters ϕ include the parameters of each neural network as well as Σ_j and transition probabilities a_{ij}. Unlike the linear PHMM it is not possible to maximize Q with respect to ϕ analytically. Instead we rely on the "generalized expectation-maximization" (GEM) algorithm in which Q is numerically maximized in the maximization step using optimization techniques. The expectation step is the same as in the linear parametric and standard HMM formulations (the forward/backward algorithm).

Gradient ascent may be used to update the network parameters in each maximization step of the GEM algorithm. When applied to multi-layer neural networks, gradient ascent (or gradient descent when the goal is to minimize "error") is often referred to as the backpropagation algorithm [2].

Rather than reiterate the development of the backpropagation algorithm here, we note that to optimize $P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta})$ via gradient ascent we use

$$\frac{\partial}{\partial \phi} \log P(\mathbf{x}_t \mid q_t = j, \boldsymbol{\theta}) = -\Sigma_j^{-1} (\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})) \quad (14)$$

where $\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})$ is the usual error quantity to be minimized by the backpropagation algorithm. Accordingly,

to maximize Q we use the backpropagation algorithm to minimize the error

$$\gamma_{tj}\Sigma_j^{-1}(\mathbf{x}_{kt} - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})) \quad (15)$$

for the network corresponding to state j.

In each maximization step of the GEM algorithm, it is not necessary to maximize Q completely. As long as Q is increased for every maximization step, the GEM algorithm is guaranteed to converge to a local maximum in the same manner as EM. In fact, since the functional Q changes with every expectation step, a complete maximization of Q in the maximization step is probably computationally wasteful. In our testing we run the gradient ascent algorithm (backpropagation algorithm) a fixed number of iterations for each GEM iteration.

4.3 Testing

In testing we desire the value of $\boldsymbol{\theta}$ which maximizes the probability of the observation sequence. Again an EM algorithm to compute $\boldsymbol{\theta}$ is appropriate.

As in the training phase, we can not maximize Q analytically, and so a GEM algorithm is necessary. To optimize Q, we use a gradient ascent algorithm:

$$\frac{\partial Q}{\partial \boldsymbol{\theta}} = \sum_t \sum_j \gamma_{tj}(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta}))^T \Sigma_j^{-1} \frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \quad (16)$$

$$\frac{\partial \hat{\boldsymbol{\mu}}_j(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = W^{(2,j)}\Lambda(g'(W^{(1,j)} + b^{(1,j)}))W^{(1,j)} \quad (17)$$

where $\Lambda(\cdot)$ forms the diagonal matrix from the components of its argument, and $g'(\cdot)$ denotes the derivative of the vector-valued function that computes the logistic function of each component of its argument.

In the results presented in this paper, we use a gradient ascent algorithm with adaptive step size. In addition it was found necessary to constrain the gradient ascent step to prevent the algorithm from wandering outside the bounds of the training data, where the output of the neural networks is essentially undefined. This constraint is implemented by simply limiting any component of the step that takes the value of $\boldsymbol{\theta}$ outside the bounds of the training data, established by the minimum and maximum $\boldsymbol{\theta}$ training values.

As with the EM training algorithm of the linear parametric case, for all of our experiments less than ten GEM iterations are required.

4.4 Freedom of choice

In [14] we present an example of a pointing gesture parameterized by projection of hand position onto the plane parallel and in front of the user at the moment that the arm is fully extended. The linear PHMM works well since the projection is a linear operation over the range of angles used in the experiment.

The nonlinear variant of the PHMM introduced in the previous section is appropriate in situations in which the dependence of the state output distributions on the parameters $\boldsymbol{\theta}$ is not linear, and cannot be made linear easily with a known coordinate transformation of the feature space.

In practice, a useful consequence of nonlinear modeling for PHMMs is that the parameter space may be chosen more freely in relation to the observation feature space. For example, in a hand gesture recognition system, the natural feature space may be the spatial position of the hand, while a natural parameterization for a pointing gesture is the spherical coordinates of the pointing direction (see Figure 2).

However, there is no guarantee that any observation feature space will permit the PHMM to learn the parameterization. Continuing with the pointing example, the nonlinear PHMM approach will learn the smooth mapping from spherical coordinates of the point to hand position at each state unambiguously. Obviously, a feature space that does not include the x coordinate (across the body) will not be enough to capture the parameterization, while a feature space that neglects the depth away from the body may work well enough.

The mapping from parameter to observations must be smooth enough to be learned by neural networks with a reasonable number of hidden units. While in theory a 3-layer logistic neural network with sufficiently many hidden units and sufficient data is capable of computing any smooth mapping, we would like to use as few hidden units as possible and so choose our parameterization and observation feature space to give simple, learnable maps. Cross validation is probably the only practical automatic procedure to evaluate parameter/observation feature space pairings, as well as the number of hidden units in each neural network. The computational complexity of such approaches is a drawback of the nonlinear PHMM approach.

In summary, with nonlinear PHMMs we are free to choose intuitive parameterizations but we must be careful that it is possible to learn the mapping from parameters to observation features given a particular observation feature space.

5 Results

To test the performance of the nonlinear PHMM, we conducted an experiment similar to the pointing experiment of [14] but with a spherical coordinate parameterization rather than the projection onto a plane in front of the user.

We used a Polhemus motion capture system to record the position of the user's wrist at a frame rate of 30Hz. Fifty such examples were collected, each averaging 29 time samples (about 1 second) in length. Thirty of the sequences were randomly selected as the training set; the remaining 20 comprised the test set.

Before training, the value of the parameter $\boldsymbol{\theta}$ must be set for each training example, as well as for each testing example to evaluate the ability of the PHMM to recover the parameterization. We directly measured the value of $\boldsymbol{\theta}$ by finding the point at which the depth of the wrist away from the user was greatest. This point was transformed to spherical coordinates (azimuth and elevation) via the arctangent function. Figure 2 diagrams the coordinate system.

Note that for pointing gestures that are confined to a small area in front of the user (as in the experiment presented in [14]) the linear parametric HMM approach will work well enough, since for small values the tangent function is approximately linear. The pointing gestures used in the present experiment were more broad, ranging from -36 to +81 degrees elevation and -77 to +80 degrees azimuth.

An 8 state causal nonlinear PHMM was trained on the 40 training examples. To simplify training we constrained the number of hidden units of each state to be equal; note

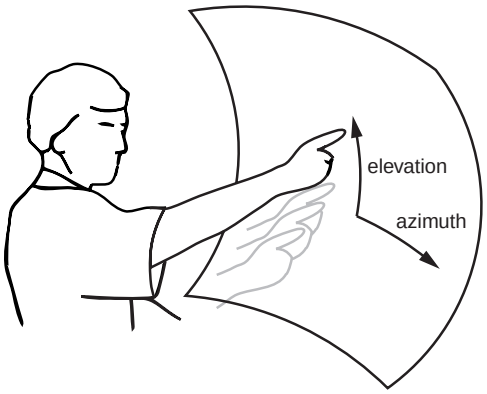

Figure 2: The spherical coordinate system is a natural parameterization of pointing direction.

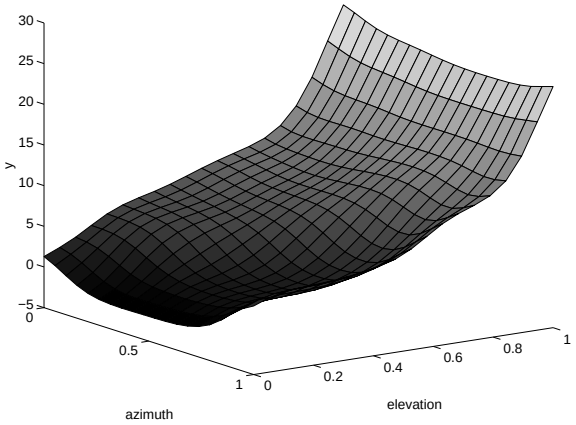

Figure 3: The output of the logistic network corresponding to state $j = 5$ displayed as a surface. State 5 is near the apex of the gesture and shows the greatest sensitivity to pointing angle. Only the y coordinate of the output is shown; the x coordinate is similarly nonlinear.

that this is not required by the model but makes choosing the number of hidden units via cross validation easier. We evaluated performance on the testing set for various numbers of hidden units and found that 10 hidden units gave the best testing performance. We did not evaluate the performance under varying amounts of training data or varying numbers of states in the HMM.

The average error over the testing set was computed to be about 6.0 degrees elevation and 7.5 degrees azimuth. Inspection of the surfaces learned by the logistic networks of the nonlinear PHMM reveals that as in the linear case, the input's dependence on θ is most dramatic in the middle of the sequence, the apex of the pointing gestures. The surface learned by the logistic network at the state corresponding to the apex captures the nonlinearity of the dependency (see Figure 3). For comparison, an eight state *linear* PHMM was trained on the same data and yielded an average error over the same test set of about 14.9 degrees elevation and 18.3 degrees azimuth.

Lastly, we demonstrate recognition performance of the nonlinear PHMM on our pointing data. A one minute sequence was collected that contained a variety of movements including six points distributed throughout. To simultaneously detect the gesture and recover θ, we used a 30 sample (one second) window on the sequence. Figure 4 shows the log probability as a function of time and the value of θ recovered for a number of recovered pointing gestures. All of the pointing gestures were recovered.

6 Conclusion

The PHMM framework presented in [14] has been generalized to handle nonlinear dependencies of the state output distributions on the parameterization θ. We have shown that where the linear PHMM employs the EM algorithm in training and testing, the nonlinear variant similarly uses the GEM algorithm.

The drawbacks of the of the generalized approach are two-fold: the number of hidden units for the networks must be chosen appropriately during training, and secondly, during testing the GEM algorithm is more computationally intensive than the EM algorithm of the linear approach.

The nonlinear PHMM is able to model a much larger class of parameterized gestures and movements than the linear parametric HMM. A benefit of the increased modeling ability is that with some care, the parameter space may be chosen independently of the observation feature space. It follows that the parameterization may be tailored to a specific gesture. Furthermore, more intuitive parameterizations may be used. For example, a family of movements may be parameterized by a subjective quantity (e.g. the "intensity" of a walk). We believe these are significant advantages in modeling parameterized gesture and movement.

References

[1] Y. Bengio and P. Frasconi. An input output HMM architecture. In G. Tesauro, M. D. S. Touretzky, and T. K. Leen, editors, *Advances in neural information processing systems 7*, pages 427–434. MIT Press, 1995.

[2] C. M. Bishop. *Neural networks for pattern recognition*. Clarendon Press, Oxford, 1995.

[3] C. M. Bishop, M. Svensen, and C. K. I. Williams. EM optimization of latent-variable density models. In M. C. Moser D. S. Touretzky and M. E. Hasselmo, editors, *Advances in neural information processing systems 8*, pages 402–408. MIT Press, 1996.

[4] C. Bregler and S. M. Omohundro. Surface learning with applications to lipreading. *Advances in neural information processing systems 6*, pages 43–50, 1994.

[5] L. W. Campbell, D. A. Becker, A. J. Azarbayejani, A. F. Bobick, and A. Pentland. Invariant features for 3-d gesture recognition. In *Second International Conference on Face and Gesture Recognition*, pages 157–162, Killington VT, 1996.

[6] R.E. Kahn and M.J. Swain. Understanding people pointing: The Perseus system. In *Proc. IEEE Int'l. Symp. on Comp. Vis.*, pages 569–574, Coral Gables, Florida, November 1995.

[7] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. *Int. J. of Comp. Vis.*, 14:5–24, 1995.

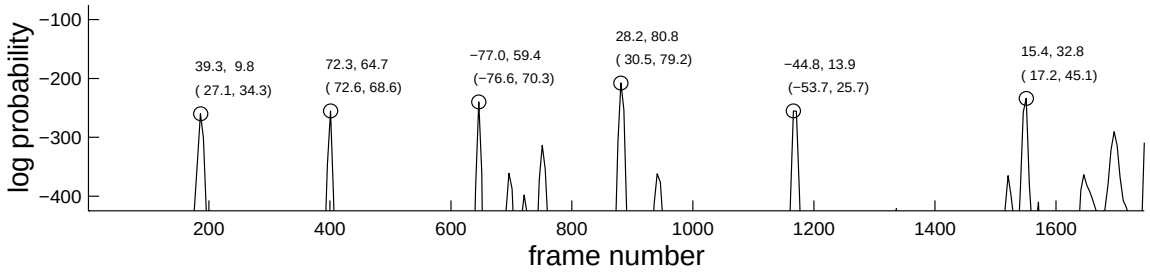

Figure 4: Recognition results are shown by the log probability of the windowed sequence beginning at each frame number. The true positive sequences are labeled by the value of θ recovered by the EM testing algorithm and the value computed by direct measurement (in parentheses).

[8] S. M. Omohundro. Family discovery. In D. S. Touretzky, M. C. Moser, and M. E. Hasselmo, editors, *Advances in neural information processing systems 8*, pages 402–408. MIT Press, 1996.

[9] H. Poizner, E. S. Klima, U. Bellugi, and R. B. Livingston. Motion analysis of grammatical processes in a visual-gestural language. In *ACM SIGGRAPH/SIGART Interdisciplinary Workshop, Motion: Representation and Perception*, pages 148–171, Toronto, April 1983.

[10] L. R. Rabiner and B. H. Juang. *Fundamentals of speech recognition*. Prentice Hall, Englewood Cliffs, 1993.

[11] J. Schlenzig, E. Hunter, and R. Jain. Vision based hand gesture interpretation using recursive estimation. In *Proc. of the Twenty-Eighth Asilomar Conf. on Signals, Systems and Comp.*, October 1994.

[12] T. E. Starner and A. Pentland. Visual recognition of American Sign Language using hidden Markov models. In *Proc. of the Intl. Workshop on Automatic Face- and Gesture-Recognition*, Zurich, 1995.

[13] A. D. Wilson and A. F. Bobick. Learning visual behavior for gesture analysis. In *Proc. IEEE Int'l. Symp. on Comp. Vis.*, Coral Gables, Florida, November 1995.

[14] A. D. Wilson and A. F. Bobick. Recognition and interpretation of parametric gesture. *Proc. Int. Conf. Comp. Vis.*, 1998. accepted for publication (see MIT Perceptual Computing Group Technical Report 421, http://www-white.media.mit.edu/vismod).

[15] A. D. Wilson, A. F. Bobick, and J. Cassell. Temporal classification of natural gesture and application to video coding. *Proc. Comp. Vis. and Pattern Rec.*, pages 948–954, 1997.