M.I.T. Media Laboratory Perceptual Computing Section Technical Report No. 424

Nonlinear Parametric Hidden Markov Models

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 [4], we extended 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 HMM was motivated by the task of simultaneoiusly recognizing and interpreting gestures that exhibit meaningful variation. With standard HMMs, such global variation confounds the recognition process. In this paper we extend the parametric HMM approach to handle nonlinear (non-analytic) dependencies of the output distributions on the parameter of interest. We show a generalized expectation-maximization (GEM) algorithm for training the parametric HMM 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 [4] we introduce parametric hidden Markov models (HMMs) as a technique to simultaneously recognize and interpret *parametric gesture*. By parametric 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 paramterized by two values: the Cartesian coordinates that indicate direction. Alternatively, direction can be specified by spherical coordinates.

We refer the reader to [4] for a detailed motivation of the parameteric HMM 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 parametric gestures faces the difficulty that the gesture can only be recognized once the value of the parameters that determine the form of the gesture are recovered, and likewise the process of recovering the parameters necessarily involves recognizing the gesture.

Parametric HMMs extend the standard HMM model to include a global parametric variation in the output of the HMM states. In [4] a linear model was used to model the parametric variation 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 parametric HMM simultaneously recognizes the gesture and estimates the quantifying parameters, also by an EM procedure.

In this paper the parametric HMM approach is extended to handle situations in which the dependence of the state output distibutions on the parameters is not linear. Nonlinear parametric HMMs accordingly model the dependence using a single 3-layer logistic neural network at each state. Before presenting nonlinear parametric HMMs in full we reiterate the mathematical developement of linear parameteric HMMs.

2 Linear parameteric hidden Markov models

2.1 Model

Parametric HMMs model the dependence on the parameter of interest explicitly. We begin with the usual HMM formulation [3] and change the form of the output probability distribution (usually a normal distribution or a mixture model) to depend on the parameter θ, a vector quantity.

In the standard continuous HMM model, a sequence is represented by movement through a set of hidden states. The Markovian 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 **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}, μ_j, and Σ_j.[1]

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

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

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

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

Note that θ is constant for the entire observation sequence, but is free to vary from sequence to sequence. When necessary, we write the value of θ associated with a particular sequence k as θ_k.

2.2 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 theta. The Baum-Welch form of the expectation-maximization (EM) algorithm is used to update the

[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.

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 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 $Q(\phi' \mid \phi)$. ϕ' may contain all the parameters in ϕ, or only a subset if several maximization steps are required to estimate all the parameters. As explained in the appendix, Q is the expected value of the log probability given the parse γ_{tj}. In the appendix we derive the derivative of Q for HMM's:

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

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

$$Z_j \equiv \begin{bmatrix} W_j \; \bar{\mu}_j \end{bmatrix} \quad \Omega_k \equiv \begin{bmatrix} \theta_k \\ 1 \end{bmatrix} \tag{4}$$

such that $\hat{\mu}_j = Z_j \Omega_k$. 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 θ_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 $\hat{\mu}_j = Z_j \Omega_k$ into equation 3:

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

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

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

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

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

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

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

as is the matrix of transition probabilities [3].

2.3 Testing

In testing we are given an HMM and an input sequence. We wish to compute the value of θ and the probability that the HMM produced the sequence. As compared to the usual HMM formulation, the parameterized HMM's testing procedure is complicated by the dependence of the parse on the unknown θ. Here we present only a technique to extract the value of θ, since for a given value of θ 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 θ 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 θ. In the corresponding maximization step we update θ to maximize Q, the log probability of the sequence given the parse γ_{tj}.

To derive an update equation for θ, we start with the derivative in equation 3 from the previous section and select θ as ϕ'. As with Z_j, only the means μ_j depend upon θ yielding:

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

Setting this derivative to zero and solving for θ, we have:

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

The values of γ_{tj} and θ are iteratively updated until the change in θ 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.

3 Non-linear parameteric hidden Markov models

3.1 Model

Nonlinear parametric hidden Markov models omit the linear model of section 2.1 in favor of a logistic neural network with one hidden layer. As with linear parametric HMM's, the output of each network is perturbed by Gaussian noise:

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

The output $\hat{\mu}_j(\theta)$ of the network associated with state j can be written as

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

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 $g(\mathbf{x})$ 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.

3.2 Training

As with standard HMMs and linear parametric HMMs, the parameters of the nonlinear parameteric HMM are updated in the maximization step of the training EM algorithm by choosing a ϕ' to maximize the auxiliary function $Q(\phi' \mid \phi)$.

In the nonlinear parametric HMM, the parameters ϕ include the parameters of each neural network as well as Σ_j and transition probabilities a_{ij}. Unlike the linear parametric HMM 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 (approximately) 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 minimize "error") is often referred to as the backpropagation algorithm [1].

Rather than reiterate the gradient descent equations for logistic neural networks here, we note that the backpropagation algorithm is appropriate for optimizing Q with the modification that the error to be "propagated" backwards into the network has the form

$$\gamma_{tj}\mathbf{\Sigma}_j^{-1}(\mathbf{x}_{kt} - \hat{\mu}_j(\theta)) \tag{14}$$

which is simply the usual error weighted by γ_{tj} and $\mathbf{\Sigma}_j^{-1}$. Intuitively, this weighting steers each network to model the appropriate part of the input, much as the gating function of a mixtures of experts model [2] selects its experts. Also, this weighting may be derived from the form of $\frac{\partial Q}{\partial \phi}$ (equation 3).

In each maximization step of the GEM algorithm, it is not necessary to completely maximize Q. As long as Q is increased for every maximization step, the GEM algorithm is guaranteed to converge to a local maximimum 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. Accordingly in our testing we run the gradient ascent algorithm (backpropagation algorithm) a fixed number of iterations for each GEM iteration.

3.3 Testing

In testing we desire the value of θ which maximizes the probability of the observation sequence. Again an EM algorithm to compute θ 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:

$$\frac{\partial Q}{\partial \theta} = \sum_t \sum_j \gamma_{tj}(\mathbf{x}_i - \hat{\mu}_j(\theta))^T \mathbf{\Sigma}_j^{-1} \frac{\partial \hat{\mu}_j(\theta)}{\partial \theta} \tag{15}$$

$$\frac{\partial \hat{\mu}_j(\theta)}{\partial \theta} = W^{(2,j)}\Lambda(g'(W^{(1,j)} + b^{(1,j)}))W^{(1,j)} \tag{16}$$

where $\Lambda(\mathbf{x})$ forms the diagonal matrix from the components of $\mathbf{x}$, and $g'(\mathbf{x})$ 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 alogirthm 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 θ outside the bounds of the training data, established by the minimum and maximum θ training values.

As with the EM training algorithm of the linear parametric case, with the examples we have tried less than ten GEM iterations are required.

4 Discussion

In [4] 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 parametric HMM approach works well since the projection is a linear operation.

The nonlinear variant of the parametric HMM introduced in the previous section is appropriate in situations in which the dependence of the state output distributions on the parameters θ is not linear, and can not be easily made linear with a known coordinate transformation of the feature space.

In practice, a useful consequence of nonlinear modeling for parametric HMMs 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.

Conversely, there is no guarantee that any observation feature space will permit the parametric HMM to learn the parameterization. Continuing with the pointing example, the nonlinear parametric HMM 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.

One difficulty in assessing whether an observation feature space and a parameter space are appropriate for one another is whether the mapping from parameter to observation features is 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 is capable of learning any mapping, we would like to use as few hidden units as possible and so choose our parameterizaiton and observation feature space to give simple, learnable mappings. 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 parameteric HMM approach.

In summary, with nonlinear parametric HMMs 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 parametric HMM, we conducted an experiment similar to the pointing experiment of [4] 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 θ must be set for each training example, as well as for each testing example to evaluate the ability of the parametric HMM to recover the parameterization. We directly measured the value of θ by finding the point at which the depth of the wrist away from the user was greatest. This point

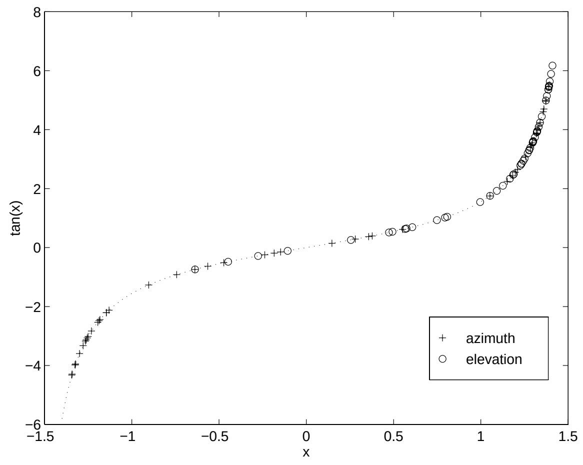

Figure 1: The distribution of θ for the pointing data sets over the tangent function.

was transformed to spherical coordinates (azimuth and elevation) via the arctangent function.

Note that for pointing gestures that are confined to a small area in front of the user (as in the experiment presented in [4]) the linear parameteric 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. Figure 1 shows how the population of θ associated with the training set is distributed over more than the nearly linear portion of the tangent function.

An eight state causal nonlinear parametric HMM was trained on the forty training examples. To simplify training we constrained the number of hidden units of each state to be equal; note that this constraint is not present in 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 output of the resulting eight neural networks is shown in Figure 3. The output of the neural networks shows how the input's dependence on θ is most dramatic in the middle of the sequence, or at the apex of the pointing gesture.

The average error over the testing set was computed to be about 6.0 degrees elevation and 7.5 degrees azimuth. For comparison, an eight state linear parametric HMM was trained on the same training data and yielded an average error over the same testing set of about 14.9 degrees elevation and 18.3 degrees azimuth.

Lastly, we demonstrate recognition performance of the nonlinear parameterized HMM 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 sec) window on the sequence. Figure 2 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 parametric hidden Markov model framework presented in [4] has been generalized to handle nonlinear dependencies of the state output distributions on the parameterization θ. We have shown that where the linear parametric HMM 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 parametric HMM is able to model a much larger class of parametric gestures and movements than the linear parametric HMM. A practical benefit of the increased modeling ability is that with some care, the parameter space may be chosen independently of the observation feature space. Theoretically, this should allow more intuitive parameterizations, including perhaps those derived from more subjective qualities of the signal (e.g. the "intensity" of a walk). Additionally, it should be easier to tailor parameter spaces to specific gestures, with all gestures employing the same observation feature space that is indiginous to the sensors. We believe that these are signicant advantages in modeling parameteric gesture and movement.

References

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

[2] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

[3] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[4] 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).

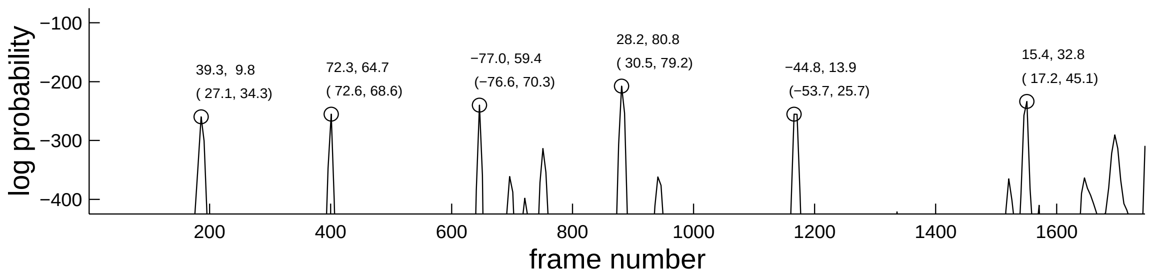

Figure 2: Recognition results are shown by the log probability of the windowed sequence beginning at each frame number. The true positive sequences are labeled by the value of θ recovered by the EM testing algorithm and the value computed by direct measurement (in parantheses).

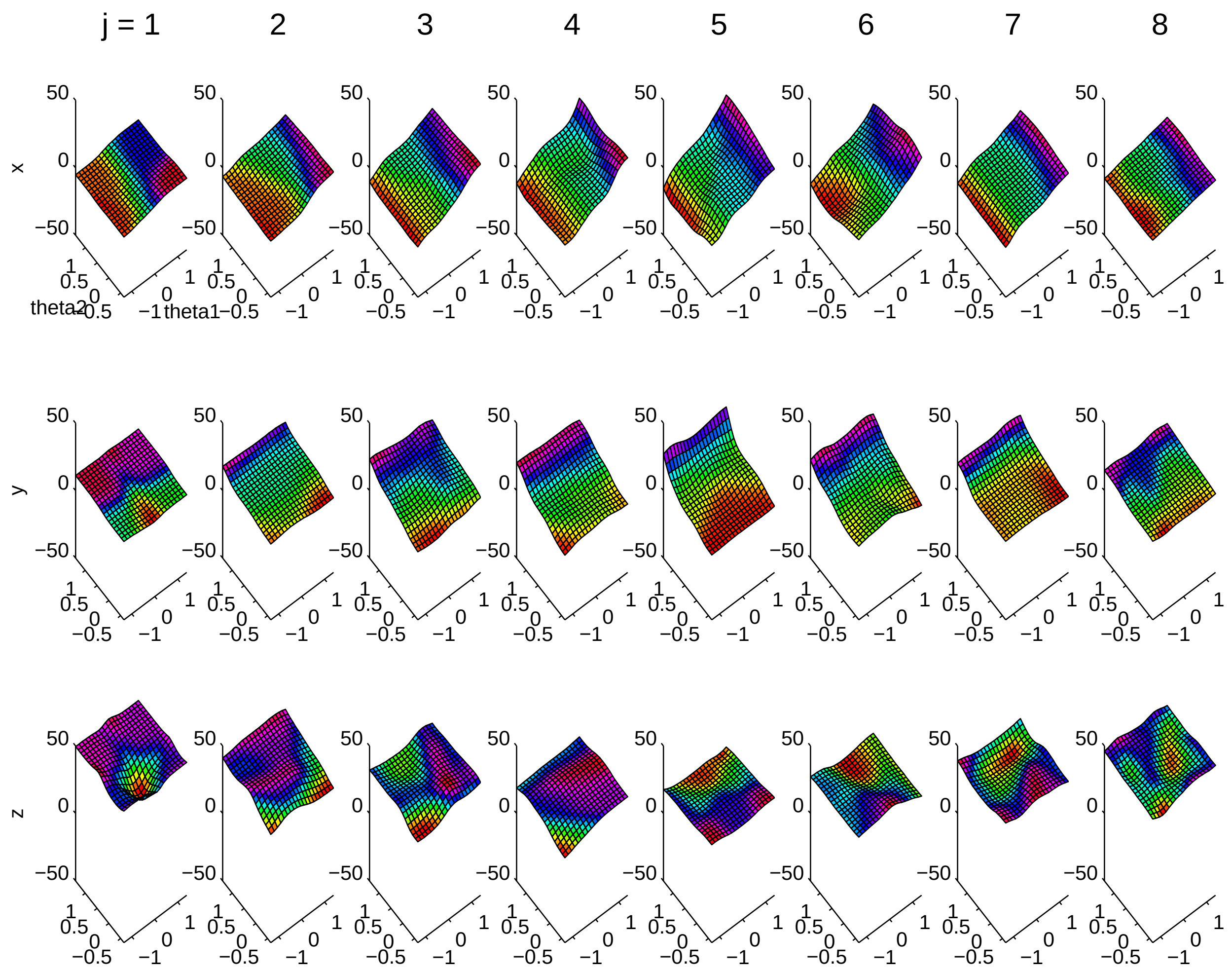

Figure 3: The output of each network is displayed (as a surface) for each of the observation feature coordinates (x, y, and z), for each of the eight states of the trained parametric HMM.