Proceedings of 1993 International Joint Conference on Neural Networks

Hierarchical mixtures of experts and the EM algorithm

Michael I. Jordan
Department of Brain and Cognitive Sciences
MIT
Cambridge, MA 02139

Robert A. Jacobs
Department of Psychology
University of Rochester
Rochester, NY 14627

Abstract

We present a tree-structured architecture for supervised learning. The statistical model underlying the architecture is a hierarchical mixture model in which both the mixture coefficients and the mixture components are generalized linear models (GLIM's). Learning is treated as a maximum likelihood problem; in particular, we present an Expectation-Maximization (EM) algorithm for adjusting the parameters of the architecture. We also develop an on-line learning algorithm in which the parameters are updated incrementally. Comparative simulation results are presented in the robot dynamics domain.

1 INTRODUCTION

In the statistical literature and in the machine learning literature, divide-and-conquer algorithms have become increasingly popular. The CART algorithm [1], the MARS algorithm [5], and the ID3 algorithm [12] are well-known examples. These algorithms fit surfaces to data by explicitly dividing the input space into a nested sequence of regions, and by fitting simple surfaces (e.g., constant functions) within these regions. The advantages of these algorithms include the interpretability of their solutions and the speed of the training process.

In this paper we present a neural network architecture that is a close cousin to architectures such as CART and MARS. As in our earlier work [6,7], we formulate the learning problem for this architecture as a maximum likelihood problem. In the current paper we utilize the Expectation-Maximization (EM) framework to derive the learning algorithm.

2 HIERARCHICAL MIXTURES OF EXPERTS

The algorithms that we discuss in this paper are supervised learning algorithms. We explicitly address the case of regression, in which the input vectors are elements of $\Re^m$ and the output vectors are elements of $\Re^n$. Our model also handles classification problems and counting problems in which the outputs are integer-valued. The data are assumed to form a countable set of paired observations $\mathcal{X} = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}$. In the case of the *batch* algorithm discussed below, this set is assumed to be finite; in the case of the *on-line* algorithm, the set may be infinite.

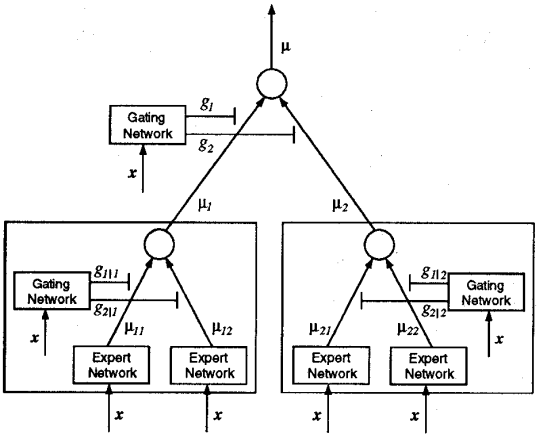

Figure 1: A two-level hierarchical mixture of experts.

We propose to solve nonlinear supervised learning problems by dividing the input space into a nested set of regions and fitting simple surfaces to the data that fall in these regions. The regions have "soft" boundaries, meaning that data points may lie simultaneously in multiple regions. The boundaries between regions are themselves simple parameterized surfaces that are adjusted by the learning algorithm.

The hierarchical mixture-of-experts (HME) architecture is shown in Figure 1.[1] The architecture is a tree in which the *gating networks* sit at the nonterminals of the tree. These networks receive the vector $\mathbf{x}$ as input and produce scalar outputs that

[1] To simplify the presentation, we restrict ourselves to a two-level hierarchy throughout the paper. All of the algorithms that we describe, however, generalize readily to hierarchies of arbitrary depth. See [9] for a recursive formalism that handles arbitrary trees.

are a partition of unity at each point in the input space. The *expert networks* sit at the leaves of the tree. Each expert produces an output vector $\boldsymbol{\mu}_{ij}$ for each input vector. These output vectors proceed up the tree, being multiplied by the gating network outputs and summed at the nonterminals.

All of the expert networks in the tree are linear with a single output nonlinearity. We will refer to such a network as "generalized linear," borrowing the terminology from statistics [11]. Expert network (i,j) produces its output $\boldsymbol{\mu}_{ij}$ as a generalized linear function of the input $\mathbf{x}$:

$$\boldsymbol{\mu}_{ij} = f(U_{ij}\mathbf{x}), \tag{1}$$

where U_{ij} is a weight matrix and f is a fixed continuous nonlinearity. The vector $\mathbf{x}$ is assumed to include a fixed component of one to allow for an intercept term.

For regression problems, $f(\cdot)$ is the identity function (i.e., the experts are linear). For binary classification problems, $f(\cdot)$ is generally taken to be the logistic function, in which case the expert outputs are interpreted as the log odds of "success" under a Bernoulli probability model. Other models (e.g., multiway classification, counting, rate estimation and survival estimation) are handled readily by making other choices for $f(\cdot)$. These models are smoothed piecewise analogs of the corresponding generalized linear models (GLIM's; cf. [11]).

The gating networks are also generalized linear. At the top level, define linear predictors ξ_i as follows:

$$\xi_i = \mathbf{v}_i^T\mathbf{x}, \tag{2}$$

where $\mathbf{v}_i$ is a weight vector. Then the i^{th} output of the top-level gating network is the "softmax" function of the ξ_i [2,11]:

$$g_i = \frac{e^{\xi_i}}{\sum_k e^{\xi_k}}. \tag{3}$$

Note that the g_i are positive and sum to one for each $\mathbf{x}$. The gating networks at the lower level are defined similarly, yielding outputs $g_{j|i}$ that are obtained by taking the softmax function of linear predictors $\xi_{ij} = \mathbf{v}_{ij}^T\mathbf{x}$.

The output vector at each nonterminal of the tree is the weighted output of the experts below that nonterminal. That is, the output at the i^{th} nonterminal in the second layer of the two-level tree is:

$$\boldsymbol{\mu}_i = \sum_j g_{j|i}\boldsymbol{\mu}_{ij}$$

and the output at the top level of the tree is:

$$\boldsymbol{\mu} = \sum_i g_i\boldsymbol{\mu}_i.$$

Note that both the g's and the $\boldsymbol{\mu}$'s depend on the input $\mathbf{x}$, thus the total output is a nonlinear function of the input.

2.1 A PROBABILITY MODEL

The hierarchy can be given a probabilistic interpretation. We suppose that the mechanism by which data are generated by the environment involves a nested sequence of decisions that terminates in a regressive process that maps $\mathbf{x}$ to $\mathbf{y}$. The decisions are modeled as multinomial random variables. That is, for each $\mathbf{x}$, we interpret the values $g_i(\mathbf{x},\mathbf{v}_i^0)$ as the multinomial probabilities associated with the first decision and the $g_{j|i}(\mathbf{x},\mathbf{v}_{ij}^0)$ as the (conditional) multinomial probabilities associated with the second decision. We use a statistical model to model these probabilities; in particular, our choice of parameterization (cf. Eqs. 2 and 3) corresponds to a *log-linear* probability model (see [8]). A log-linear model is a special case of a GLIM that is commonly used for "soft" multiway classification [11]. Under the log-linear model, we interpret the gating networks as modeling the input-dependent, multinomial probabilities of making particular nested sequences of decisions.

Once a particular sequence of decisions has been made, output $\mathbf{y}$ is assumed to be generated according to the following generalized linear statistical model. First, a linear predictor $\boldsymbol{\eta}_{ij}$ is formed:

$$\boldsymbol{\eta}_{ij}^0 = U_{ij}^0\mathbf{x},$$

where the superscript refers to the "true" values of the parameters. The expected value of $\mathbf{y}$ is obtained by passing the linear predictor through the *link function* f:

$$\boldsymbol{\mu}_{ij}^0 = f(\boldsymbol{\eta}_{ij}^0).$$

The output $\mathbf{y}$ is then chosen from a probability density P, with mean $\boldsymbol{\mu}_{ij}^0$ and "dispersion" parameter $\boldsymbol{\phi}_{ij}^0$. We denote the density of $\mathbf{y}$ as:

$$P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}_{ij}^0),$$

where the parameter vector $\boldsymbol{\theta}_{ij}^0$ includes the weights U_{ij}^0 and the dispersion parameter $\boldsymbol{\phi}_{ij}^0$. We assume the density P to be a member of the exponential family of densities [11]. The interpretation of the dispersion parameter depends on the particular choice of density. For example, in the case of the n-dimensional Gaussian, the dispersion parameter is the covariance matrix Σ_{ij}^0.[2]

Given these assumptions, the total probability of generating $\mathbf{y}$ from $\mathbf{x}$ is the mixture of the probabilities of generating $\mathbf{y}$ from each of the component densities, where the mixture components are multinomial probabilities:

$$P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}^0) = \sum_i g_i(\mathbf{x},\mathbf{v}_i^0)\sum_j g_{j|i}(\mathbf{x},\mathbf{v}_{ij}^0)P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}_{ij}^0). \tag{4}$$

[2] Not all exponential family densities have a dispersion parameter; in particular, the Bernoulli density has no dispersion parameter.

Note that θ^0 includes the expert network parameters ${\theta_{ij}}^0$ as well as the gating network parameters $\mathbf{v}_i^0$ and $\mathbf{v}_{ij}^0$. Note also that we can utilize Eq. 4 without the superscripts to refer to the probability model defined by a particular HME architecture, irrespective of any reference to a "true" model.

2.2 POSTERIOR PROBABILITIES

In developing the learning algorithms to be presented in the remainder of the paper, it will prove useful to define posterior probabilities associated with the nodes of the tree. The terms "posterior" and "prior" have meaning in this context during the training of the system. We refer to the probabilities g_i and $g_{j|i}$ as *prior* probabilities, because they are computed based only on the input $\mathbf{x}$, without knowledge of the corresponding target output $\mathbf{y}$. A *posterior* probability is defined once both the input and the target output are known. Using Bayes' rule, we define the posterior probabilities at the nodes of the tree as follows:

$$h_i = \frac{g_i \sum_j g_{j|i} P_{ij}(\mathbf{y})}{\sum_i g_i \sum_j g_{j|i} P_{ij}(\mathbf{y})} \tag{5}$$

and

$$h_{j|i} = \frac{g_{j|i} P_{ij}(\mathbf{y})}{\sum_j g_{j|i} P_{ij}(\mathbf{y})}, \tag{6}$$

where we have dropped the dependence on the input and the parameters to simplify the notation.

We will also find it useful to define the joint posterior probability h_{ij}, the product of h_i and $h_{j|i}$. This quantity is the probability that expert network (i,j) can be considered to have generated the data, based on knowledge of both the input and the output. Once again, we emphasize that all of these quantities are conditional on the input $\mathbf{x}$.

In deeper trees, the posterior probability associated with an expert network is simply the product of the conditional posterior probabilities along the path from the root of the tree to that expert.

2.3 THE LIKELIHOOD

We treat the problem of learning in the HME architecture as a maximum likelihood estimation problem. The log likelihood of a data set $\mathcal{X} = \{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_1^N$ is obtained by taking the log of the product of N densities of the form of Eq. 4, which yields the following log likelihood:

$$l(\theta; \mathcal{X}) = \sum_t \ln \sum_i g_i^{(t)} \sum_j g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)}). \tag{7}$$

We wish to maximize this function with respect to the parameters θ.

2.4 THE EM ALGORITHM

In the following sections we develop a learning algorithm for the HME architecture based on the Expectation-Maximization (EM) framework [4]. We derive an EM algorithm for the architecture that consists of the iterative solution of a coupled set of iteratively-reweighted least-squares problems.

EM is an iterative approach to maximum likelihood estimation. Each iteration of an EM algorithm is composed of two steps: an Estimation (E) step and a Maximization (M) step. The M step involves the maximization of a likelihood function that is redefined in each iteration by the E step. An application of EM generally begins with the observation that the optimization of the likelihood function $l(\theta; \mathcal{X})$ would be simplified if only a set of additional variables, called "missing" or "hidden" variables, were known. In this context, we refer to the observable data $\mathcal{X}$ as the "incomplete data" and posit a "complete data" set $\mathcal{Y}$ that includes the missing variables $\mathcal{Z}$. We specify a probability model that links the fictive missing variables to the actual data: $P(\mathbf{y}, \mathbf{z}|\mathbf{x}, \theta)$. The logarithm of the density P defines the "complete-data likelihood," $l_c(\theta; \mathcal{Y})$. The original likelihood, $l(\theta; \mathcal{X})$, is referred to in this context as the "incomplete-data likelihood." It is the relationship between these two likelihood functions that motivates the EM algorithm. Note that the complete-data likelihood is a random variable, because the missing variables $\mathcal{Z}$ are in fact unknown. An EM algorithm first finds the expected value of the complete-data likelihood, given the observed data and the current model. This is the E step:

$$Q(\theta, \theta^{(p)}) = E[l_c(\theta; \mathcal{Y})|\mathcal{X}],$$

where $\theta^{(p)}$ is the value of the parameters at the p^{th} iteration and the expectation is taken with respect to $\theta^{(p)}$. This step yields a deterministic function Q. The M step maximizes this function with respect to θ to find the new parameter estimates $\theta^{(p+1)}$:

$$\theta^{(p+1)} = \arg\max_{\theta} Q(\theta, \theta^{(p)}).$$

The E step is then repeated to yield an improved estimate of the complete likelihood and the process iterates.

An iterative step of EM chooses a parameter value that increases the value of Q, the expectation of the complete likelihood. What is the effect of such a step on the incomplete likelihood? Dempster, et al. proved that an increase in Q implies an increase in the incomplete likelihood:

$$l(\theta^{(p+1)}; \mathcal{X}) \geq l(\theta^{(p)}; \mathcal{X}).$$

with equality obtaining only at the stationary points of l. Thus the likelihood l increases monotonically

along the sequence of parameter estimates generated by an EM algorithm. In practice this implies convergence to a local maximum.

2.5 APPLYING EM TO THE HME ARCHITECTURE

To develop an EM algorithm for the HME architecture, we must define appropriate "missing data" so as to simplify the likelihood function. We define indicator variables z_{ij} such that one and only one of the z_{ij} is one for any given data point. These indicator variables have the interpretation as labels that specify which expert in the probability model generated the data point. This choice of missing data yields the following complete-data likelihood:

$$l_c(\theta; \mathcal{X}) = \sum_t \sum_i \sum_j z_{ij}^{(t)} \ln\{g_i^{(t)} g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)})\}. \quad (8)$$

Note the relationship of the complete-data likelihood in Eq. 8 to the incomplete-data likelihood in Eq. 7. The use of the indicator variables z_{ij} has allowed the logarithm to be brought inside the summation signs, substantially simplifying the maximization problem. We now define the E step of the EM algorithm by taking the expectation of the complete-data likelihood:

$$Q(\theta, \theta^{(p)}) = \sum_t \sum_i \sum_j h_{ij}^{(t)} \ln\{g_i^{(t)} g_{j|i}^{(t)} P_{ij}(\mathbf{y}^{(t)})\}, \quad (9)$$

where we have used the fact that:

$$\begin{aligned} E[z_{ij}^{(t)}] &= P(z_{ij}^{(t)} = 1 | \mathbf{y}^{(t)}, \mathbf{x}^{(t)}, \theta^{(p)}) \\ &= \frac{P(\mathbf{y}^{(t)} | z_{ij}^{(t)} = 1, \mathbf{x}^{(t)}, \theta^{(p)}) P(z_{ij}^{(t)} = 1 | \mathbf{x}^{(t)}, \theta^{(p)})}{P(\mathbf{y}^{(t)} | \mathbf{x}^{(t)}, \theta^{(p)})} \\ &= \frac{P(\mathbf{y}^{(t)} | \mathbf{x}^{(t)}, \theta_{ij}^{(p)}) g_i^{(t)} g_{j|i}^{(t)}}{\sum_i g_i^{(t)} \sum_j g_{j|i}^{(t)} P(\mathbf{y}^{(t)} | \mathbf{x}^{(t)}, \theta_{ij}^{(p)})} \\ &= h_{ij}^{(t)}. \end{aligned}$$

The M step requires maximizing $Q(\theta, \theta^{(p)})$ with respect to the expert network parameters and the gating network parameters. Examining Eq. 9, we see that the expert network parameters influence the Q function only through the terms $h_{ij}^{(t)} \ln P_{ij}(\mathbf{y}^{(t)})$, and the gating network parameters influence the Q function only through the terms $h_{ij}^{(t)} \ln g_i^{(t)}$ and $h_{ij}^{(t)} \ln g_{j|i}^{(t)}$. Thus the M step involves the following separate maximizations:

$$\theta_{ij}^{(p+1)} = \arg\max_{\theta_{ij}} \sum_t h_{ij}^{(t)} \ln P_{ij}(\mathbf{y}^{(t)}),$$

$$\mathbf{v}_i^{(p+1)} = \arg\max_{\mathbf{v}_i} \sum_t \sum_k h_k^{(t)} \ln g_k^{(t)},$$

and

$$\mathbf{v}_{ij}^{(p+1)} = \arg\max_{\mathbf{v}_{ij}} \sum_t \sum_k h_k^{(t)} \sum_l h_{l|k}^{(t)} \ln g_{l|k}^{(t)}.$$

Note that each of these maximization problems are themselves maximum likelihood problems, given that P_{ij}, g_i and $g_{j|i}$ are probability densities. Moreover, given our parameterization of these densities, the log likelihoods that we have obtained are weighted log likelihoods for generalized linear models (GLIM's). An efficient algorithm known as iteratively reweighted least-squares (IRLS) is available to solve the maximum likelihood problem for such models [11]. (See [8] for a discussion of IRLS.)

In summary, the EM algorithm that we have obtained involves a calculation of posterior probabilities in the outer loop (the E step), and the solution of a set of weighted IRLS problems in the inner loop (the M step). We summarize the algorithm as follows:

HME Algorithm 1

1. For each data pair $(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})$, compute the posterior probabilities $h_i^{(t)}$ and $h_{j|i}^{(t)}$ using the current values of the parameters.
2. For each expert (i, j), solve an IRLS problem with observations $\{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)})\}_1^N$ and observation weights $\{h_{ij}^{(t)}\}_1^N$.
3. For each top-level gating network, solve an IRLS problem with observations $\{(\mathbf{x}^{(t)}, h_k^{(t)})\}_1^N$.
4. For each lower-level gating network, solve an IRLS problem with observations $\{(\mathbf{x}^{(t)}, h_{l|k}^{(t)})\}_1^N$ and observation weights $\{h_k^{(t)}\}_1^N$.
5. Iterate using the updated parameter values.

---

[8] also presents an approximation to this algorithm in which the gating networks are fit by least-squares rather than maximum likelihood. In this case, the IRLS inner loop reduces to a weighted least-squares problem that can be solved without iteration.

2.5.1 Simulation Results

We tested the algorithm on a nonlinear system identification problem. The data were obtained from a simulation of a four-joint robot arm moving in three-dimensional space. The network must learn the *forward dynamics* of the arm; a mapping from twelve coupled input variables to four output variables. This mapping is rather smooth and we expect

Architecture	Relative Error	# Epochs
linear	.31	1
backprop	.09	5,500
HME	.10	35
CART	.17	NA
CART (oblique)	.13	NA
MARS	.16	NA

Table 1: Average Values of Relative Error and Number of Epochs Required for Convergence for the Batch Algorithms.

the error for a global fitting algorithm like backpropagation to be small; our main interest is in the training time.

We generated 15,000 data points for training and 5,000 points for testing. For each epoch (i.e., each pass through the training set), we computed the relative error on the test set. Relative error is computed as a ratio between the mean squared error and the mean squared error that would be obtained if the learner were to output the mean value of the outputs for all data points.

We compared the performance of a binary hierarchy to that of the best linear approximation, a backpropagation network, the CART algorithm and the MARS algorithm. The hierarchy was a four-level hierarchy with 16 expert networks and 15 gating networks. Each expert network had 4 output units and each gating network had 1 output unit. The backpropagation network had 60 hidden units, which yields approximately the same number of parameters in the network as in the hierarchy. The MARS algorithm was run with a maximum of 16 basis functions, based on the fact that each such function corresponds roughly to a single expert in the HME architecture.

Table 1 reports the average values of minimum relative error and the convergence times for all architectures. As can be seen in the Table, the backpropagation algorithm required 5,500 passes through the data to converge to a relative error of 0.09. The HME algorithm converged to a similar relative error in only 35 passes through the data. CART and MARS required similar CPU time as compared to the HME algorithm, but produced less accurate fits. (For further details on the simulation, see [8]).

As shown in Figure 2, the HME architecture lends itself well to graphical investigation. This figure displays the time sequence of the distributions of posterior probabilities across the training set at each node of the tree. At Epoch 0, before any learning has taken place, most of the posterior probabilities at each node are approximately 0.5 across the training set. As the training proceeds, the histograms flatten

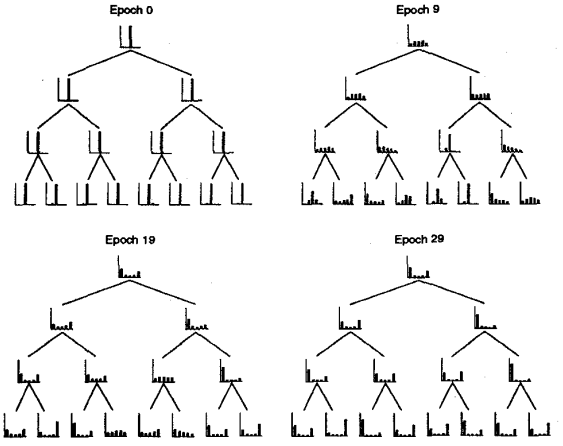

Figure 2: A sequence of histogram trees for the HME architecture. Each histogram displays the distribution of posterior probabilities across the training set at each node in the tree.

out, eventually approaching bimodal distributions in which the posterior probabilities are either one or zero for most of the training patterns. This evolution is indicative of increasingly sharp splits being fit by the gating networks. Note that there is a tendency for the splits to be formed more rapidly at higher levels in the tree than at lower levels.

2.6 AN ON-LINE ALGORITHM

Jordan and Jacobs [8] derive an on-line algorithm for the HME architecture using techniques from recursive estimation theory [10].

The on-line update rule for the parameters of the expert networks is given by the following recursive equation:

$$U_{ij}^{(t+1)} = U_{ij}^{(t)} + h_i^{(t)} h_{j|i}^{(t)} (\mathbf{y}^{(t)} - \boldsymbol{\mu}_{ij}^{(t)}) \mathbf{x}^{(t)T} R_{ij}^{(t)}, \quad (10)$$

where R_{ij} is the inverse covariance matrix for expert network (i, j). This matrix is updated via the equation:

$$R_{ij}^{(t)} = \lambda^{-1} R_{ij}^{(t-1)} - \lambda^{-1} \frac{R_{ij}^{(t-1)} \mathbf{x}^{(t)} \mathbf{x}^{(t)T} R_{ij}^{(t-1)}}{\lambda [h_{ij}^{(t)}]^{-1} + \mathbf{x}^{(t)T} R_{ij}^{(t-1)} \mathbf{x}^{(t)}}, \quad (11)$$

where λ is a decay parameter.

Similar update rules are obtained for the parameters of the gating networks. See [8] for further details.

2.6.1 Simulation Results

The on-line algorithm was tested on the robot dynamics problem described in the previous section.

Architecture	Relative Error	# Epochs
linear	.32	1
backprop (on-line)	.08	63
HME (on-line)	.12	2

Table 2: Average values of relative error and number of epochs required for convergence for the on-line algorithms.

The performance of the algorithm was compared to an on-line backpropagation network.

The minimum values of relative error and the convergence times for both architectures are provided in Table 2.

The on-line algorithm for backpropagation is significantly faster than the corresponding batch algorithm (cf. Table 1). This is also true of the on-line HME algorithm, which converges in two passes through the data.

3 CONCLUSIONS

We have presented a novel tree-structured architecture for supervised learning. This architecture is based on a statistical model, and makes contact with a number of branches of statistical theory, including mixture model estimation and generalized linear model theory. The learning algorithm for the architecture is an EM algorithm.

The major advantage of the HME approach over related decision tree and multivariate spline algorithms such as CART, MARS and ID3 is the use of a statistical framework. The statistical framework motivates some of the variance-decreasing features of the HME approach, such as the use of "soft" boundaries. The statistical approach also provides a unified framework for handling a variety of data types, including binary variables, ordered and unordered categorical variables, and real variables, both at the input and the output. The use of maximum likelihood allows standard tools from statistical theory to be brought to bear in developing inference procedures, fitting procedures and measures of uncertainty for the architecture. It also opens the door to the Bayesian approaches that have been found to be useful in the context of unsupervised mixture model estimation [3].

Acknowledgements

This project was supported by grants from the McDonnell-Pew Foundation, ATR Auditory and Visual Perception Research Laboratories, Siemens Corporation, by grant IRI-9013991 from the National Science Foundation, and by grant N00014-90-J-1942 from the Office of Naval Research. Michael I. Jordan is a NSF Presidential Young Investigator.

References

[1] Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group.

[2] Bridle, J. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fogelman–Soulie & J. Hérault (Eds.), *Neurocomputing: Algorithms, Architectures, and Applications.* New York: Springer-Verlag.

[3] Cheeseman, P., Kelly, J., Self, M., Stutz, J., Taylor, W., & Freeman, D. (1988). Autoclass: A Bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, Ann Arbor, MI.

[4] Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, B, 39*, 1-38.

[5] Friedman, J. H. (1991). Multivariate adaptive regression splines. *The Annals of Statistics, 19*, 1-141.

[6] Jacobs, R. A, Jordan, M. I., Nowlan, S. J., & Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation, 3*, 79-87.

[7] Jordan, M. I. & Jacobs, R. A. (1992). Hierarchies of adaptive experts. In J. Moody, S. Hanson, & R. Lippmann (Eds.), *Advances in Neural Information Processing Systems 4.* San Mateo, CA: Morgan Kaufmann.

[8] Jordan, M. I. & Jacobs, R. A. (1993). *Hierarchical mixtures of experts and the EM algorithm.* Computational Cognitive Science Tech. Rep. 9301, MIT, Cambridge, MA.

[9] Jordan, M. I. & Xu, L. (1993). *Convergence results for the EM approach to mixtures-of-experts architectures.* Computational Cognitive Science Tech. Rep. 9303, MIT, Cambridge, MA.

[10] Ljung, L. & Söderström, T. (1986). *Theory and practice of recursive identification.* Cambridge: MIT Press.

[11] McCullagh, P. & Nelder, J.A. (1983). *Generalized Linear Models.* London: Chapman and Hall.

[12] Quinlan, R. (1986). Induction of decision trees. *Machine Learning, 1*, 81-106.