In press: M. Arbib (Ed.), *The Handbook of Brain Theory and Neural Networks.* Cambridge, MA: MIT Press.

Modular and hierarchical learning systems

Michael I. Jordan
Department of Electrical Engineering and Computer Science
Department of Statistics
University of California, Berkeley

Robert A. Jacobs
Department of Brain and Cognitive Sciences
University of Rochester

RUNNING HEAD: Modular and Hierarchical Systems

Correspondence:
Michael I. Jordan
EECS Computer Science Division
387 Soda Hall # 1776
Berkeley, CA 94720-1776
Phone: (510) 642-3806
Fax: (510) 642-5775
email: jordan@cs.berkeley.edu

1 Introduction

In this article we discuss the problem of learning in modular and hierarchical systems. Modular and hierarchical systems allow complex learning problems to be solved by dividing the problem into a set of sub-problems, each of which may be simpler to solve than the original problem. Within the context of supervised learning—our focus in this article—modular architectures arise when we assume that the data can be well described by a collection of functions, each of which is defined over a relatively local region of the input space. A modular architecture can model such data by allocating different modules to different regions of the space. Hierarchical architectures arise when we assume that the data are well described by a multi-resolution model—a model in which regions are divided recursively into sub-regions.

Modular and hierarchical systems present an interesting credit assignment problem—it is generally the case that the learner is not provided with prior knowledge of the partitioning of the input space (but cf. Hampshire & Waibel, 1989). Knowledge of the partition would correspond to being given "labels" specifying how to allocate modules to data points. The assumption we make is that such labels are absent. The situation is reminiscent of the unsupervised clustering problem in which a classification rule must be inferred from a data set in which the class labels are absent and, indeed, the connection to clustering has played an important role in the development of the supervised learning algorithms that we present here.

The learning algorithms that we describe below solve the credit assignment problem by computing a set of values—posterior probabilities—that can be thought of as estimates of the missing "labels." These posterior probabilities are based on a probabilistic model of the input-output characteristics of each of the network modules. This approach to learning in modular systems was developed by Jacobs, Jordan, Nowlan and Hinton (1991). Jordan and Jacobs (1994) extended the modular system to a hierarchical system, made links to the statistical literature on classification and regression trees (Breiman, Friedman, Olshen, & Stone, 1984), and developed an Expectation-Maximization (EM) algorithm for the architecture. We describe these developments in the remainder of the article, emphasizing the probabilistic framework.

2 The Mixture of Experts (ME) Architecture

The modular architecture that we consider is shown in Figure 1. The architecture is composed of N modules referred to as *expert networks*, each of which

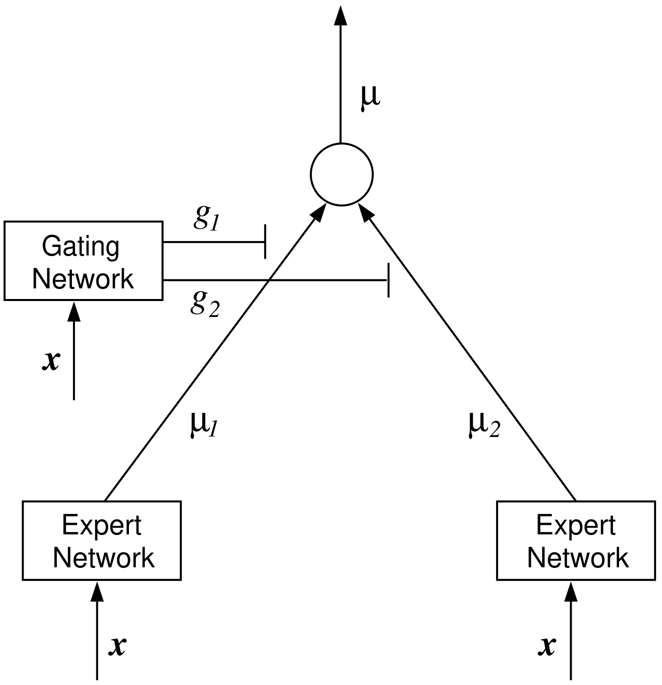

Figure 1: A mixture of experts architecture. The output $\boldsymbol{\mu}$ is the conditional mean of $\mathbf{y}$ given $\mathbf{x}$ (see text).

implements a parameterized function $\boldsymbol{\mu}_i = f(\mathbf{x}, \boldsymbol{\theta}_i)$ from inputs $\mathbf{x}$ to outputs $\boldsymbol{\mu}_i$, where $\boldsymbol{\theta}_i$ is a parameter vector. We attach a probabilistic interpretation to each of the expert networks by assuming that the experts generate outputs $\boldsymbol{y}$ with probability $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}_i)$, where $\boldsymbol{\mu}_i$ is the mean of the density P.

Because we assume that different expert networks are appropriate in different regions of the input space, the architecture requires a mechanism that identifies, for any given input $\mathbf{x}$, that expert or blend of experts that is most likely to produce the correct output. This is accomplished via an auxiliary network, known as a *gating network*, that produces as output a set of scalar coefficients g_i that serve to weight the contributions of the various experts. These coefficients are not fixed constants but vary as a function of the input $\mathbf{x}$.

The probabilistic interpretation of the gating network is as a *classifier*, a system that maps an input $\boldsymbol{x}$ into the probabilities that the various experts will be able to generate the desired output (based on knowledge of $\boldsymbol{x}$ alone). These probabilities (the g_i) are constrained to be nonnegative and sum to one (for each $\boldsymbol{x}$).

There are many ways to enforce the probabilistic constraints on the g_i. One approach is to utilize the *softmax* function (Jacobs, et al., 1991). Define an intermediate set of variables ξ_i as parameterized functions of the input $\mathbf{x}$:

$$\xi_i = \xi_i(\mathbf{x}, \boldsymbol{\eta}), \tag{1}$$

where $\boldsymbol{\eta}$ is a parameter vector, and define the outputs g_i in terms of the ξ_i as follows:

$$g_i = \frac{e^{\xi_i}}{\sum_j e^{\xi_j}}. \tag{2}$$

It is readily verified that the g_i are nonnegative and sum to one for each $\mathbf{x}$. This approach has the virtue of having a simple probabilistic interpretation: the ξ_i are the discriminant surfaces for a classification problem involving the exponential family of probability distributions (Jordan & Jacobs, 1994).

2.1 The Mixture Model

Let us now specify the probabilistic model underlying the mixture of experts architecture more precisely. We assume that the training set $\mathcal{X} = \{(\mathbf{x}^{(l)}, \mathbf{y}^{(l)})\}_{l=1}^{L}$ is generated in the following way. Given the choice of an input $\mathbf{x}$, a label i is chosen with probability $P(i|\mathbf{x}, \boldsymbol{\eta}^0)$ (where the superscript '0' distinguishes the parameter values in the model from the parameter estimates in the network). Given the choice of the label and given the input, the target output $\mathbf{y}$ is assumed to be generated with probability $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_i^0)$. Each such data point is assumed to be generated independently in this manner.

Note that a given output $\mathbf{y}$ can be generated in N different ways, corresponding to the N different choices of the label i. Thus the total probability of generating $\mathbf{y}$ from $\mathbf{x}$ is given by the sum over i:

$$P(\mathbf{y}|\mathbf{x}, \boldsymbol{\Theta}^0) = \sum_i P(i|\mathbf{x}, \boldsymbol{\eta}^0) P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_i^0), \tag{3}$$

where $\boldsymbol{\Theta}^0$ denotes the vector of all of the parameters ($\boldsymbol{\Theta}^0 = [\boldsymbol{\theta}_1^0, \boldsymbol{\theta}_2^0, \ldots, \boldsymbol{\theta}_N^0, \boldsymbol{\eta}^0]^T$). The density in Equation 3 is known as a *mixture density*. It is a mixture density in the output space, conditional on the choice of input.

It is the task of the gating network to model the probabilities $P(i|\mathbf{x}, \boldsymbol{\eta}^0)$, which can be construed as class probabilities in a multiway classification problem of the input $\boldsymbol{x}$. We parameterize these probabilities via Equations 1 and 2, identifying the gating network outputs g_i with $P(i|\mathbf{x}, \boldsymbol{\eta})$.

It is straightforward to compute moments of the mixture density. For example, the conditional mean $\boldsymbol{\mu} = E(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\Theta})$ is readily obtained by taking the expected value of Equation 3:

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

where $\boldsymbol{\mu}_i$ is the conditional mean associated with the probability distribution $P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_i^0)$. This convex combination of the outputs of the experts is a natural choice for the output of the modular architecture (cf. Figure 1), although it can also be advantageous to make fuller use of the entire mixture density.

2.2 A Gradient-Based Learning Algorithm

To develop an algorithm for estimating the parameters of a mixture of experts architecture, we make use of the maximum likelihood (ML) principle. That is, we choose parameters for which the probability of the training set given the parameters (a function known as the *likelihood*) is largest. As is commonly the case in statistics, it is actually more convenient to work with the logarithm of the likelihood than the likelihood itself. Taking the logarithm of the product of N densities of the form of Equation 3 yields the following log likelihood:

$$l(\mathcal{X}, \boldsymbol{\Theta}) = \sum_l \log \sum_i P(i|\mathbf{x}^{(l)}, \boldsymbol{\eta}) P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_i). \tag{4}$$

One approach to maximizing the log likelihood is to use gradient ascent (a better approach is to use the EM algorithm; see below). Computing the gradient of l with respect to $\boldsymbol{\mu}_i$ and ξ_i yields:

$$\frac{\partial l}{\partial \boldsymbol{\mu}_i} = \sum_l h_i^{(l)} \frac{\partial}{\partial \boldsymbol{\mu}_i} \log P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_i) \tag{5}$$

and

$$\frac{\partial l}{\partial \xi_i} = \sum_l (h_i^{(l)} - g_i^{(l)}), \tag{6}$$

where $h_i^{(l)}$ is defined as $P(i|\mathbf{x}^{(l)}, \mathbf{y}^{(l)})$. In deriving this result, we have used Bayes' rule:

$$P(i|\mathbf{x}^{(l)}, \mathbf{y}^{(l)}) = \frac{P(i|\mathbf{x}^{(l)}) P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, i)}{\sum_j P(j|\mathbf{x}^{(l)}) P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, j)}.$$

This suggests that we define $h_i^{(l)}$ as the *posterior probability* of the i^{th} label, conditional on the input $\mathbf{x}^{(l)}$ and the output $\mathbf{y}^{(l)}$. Similarly, the probability $g_i^{(l)}$ can be interpreted as the *prior probability* $P(i|\mathbf{x}^{(l)})$, the probability of the i^{th} label, given only the input $\mathbf{x}^{(l)}$. Given these definitions, Equation 6 has the natural interpretation of moving the prior probabilities toward the posterior probabilities.

An interesting special case is an architecture in which the expert networks and the gating network are linear and the probability density associated with the experts is a Gaussian with identity covariance matrix. In this case, Equations 5 and 6 yield the following on-line learning algorithm ("on-line" meaning that we have dropped the summation across l):

$$\Delta\boldsymbol{\theta}_i = \rho h_i^{(l)}(\mathbf{y}^{(l)} - \boldsymbol{\mu}_i^{(l)})\mathbf{x}^{(l)T} \tag{7}$$

and

$$\Delta\boldsymbol{\eta}_i = \rho(h_i^{(l)} - g_i^{(l)})\mathbf{x}^{(l)T}, \tag{8}$$

where ρ is a learning rate. Note that both of these equations have the form of the classical LMS rule ("delta algorithm"), with the updates for the experts in Equation 7 being modulated by their posterior probabilities.

It is also of interest to examine the expression for the posterior probability in the Gaussian case:

$$h_i^{(l)} = \frac{g_i^{(l)} e^{-\frac{1}{2}(\mathbf{y}^{(l)}-\boldsymbol{\mu}_i^{(l)})^T(\mathbf{y}^{(l)}-\boldsymbol{\mu}_i^{(l)})}}{\sum_j g_j^{(l)} e^{-\frac{1}{2}(\mathbf{y}^{(l)}-\boldsymbol{\mu}_j^{(l)})^T(\mathbf{y}^{(l)}-\boldsymbol{\mu}_j^{(l)})}}. \tag{9}$$

This is a normalized distance measure that reflects the relative magnitudes of the residuals $\mathbf{y}^{(l)} - \boldsymbol{\mu}_i^{(l)}$. If the residuals for expert i are small relative to those of the other experts then $h_i^{(l)}$ is large, otherwise $h_i^{(l)}$ is small. Note moreover that the $h_i^{(l)}$ are positive and sum to one for each $\mathbf{x}^{(l)}$; this implies that credit is distributed to the experts in a competitive manner.

It is straightforward to utilize other members of exponential family of densities as component densities for the experts, to allow dispersion (e.g., covariance) parameters to be incorporated in the model and to estimate the dispersion parameters via the learning algorithm (Jordan & Jacobs, 1994; Jordan & Xu, 1995).

3 The Hierarchical Mixture of Experts (HME) Architecture

The ME architecture solves complex function approximation problems by allocating different modules to different regions of the input space. This approach can have advantages for problems in which the modules are simpler than the large network that would be required to solve the problem as a whole. If we

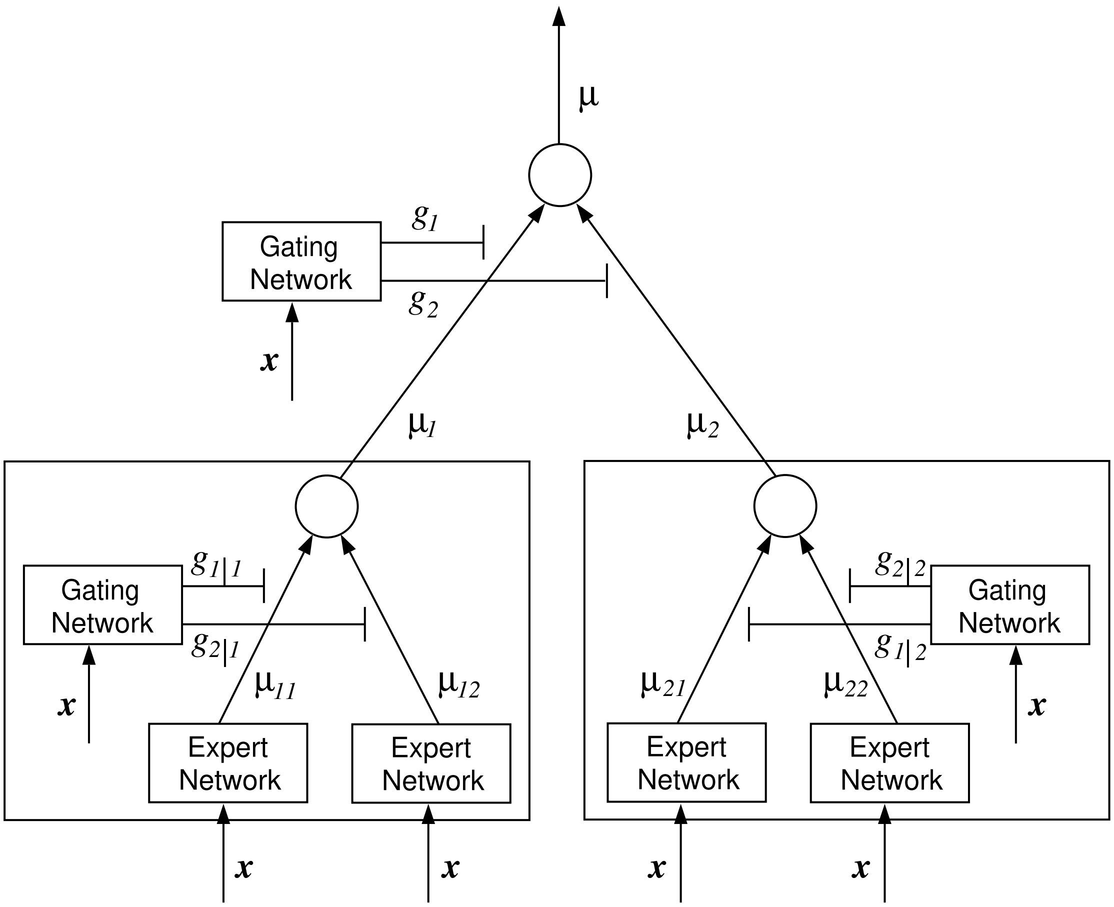

Figure 2: A two-level binary hierarchical architecture. The top-level gating network produces coefficients g_i that effectively split the input space into regions, and the lower-level gating networks produce coefficients $g_{j|i}$ that effectively split these regions into sub-regions. The expert networks fit surfaces within these nested regions. Deeper trees are formed by expanding the expert networks recursively into additional gating networks and sub-experts.

now inquire about the internal structure of a module, however, we see that the same argument can be repeated. Perhaps it is better to split a module into simpler sub-modules rather than to use a single module to fit the data in a region. This suggests a thoroughgoing divide-and-conquer approach to supervised learning in which a tree-structured architecture is used to perform multiple, nested splits of the input space (see Figure 2). The splitting process terminates in a set of expert networks at the leaves of the tree, which, because they are defined over relatively small regions of the input space, can fit simple functions to the data (e.g., linear). This hierarchical architecture, suggested by Jordan and Jacobs (1994), has close ties to the classification and regression tree models in statistics and machine learning (e.g., Breiman, et al., 1984). Indeed, the architecture can be viewed as a probabilistic variant of such models.

The mathematical framework underlying the HME architecture is essentially the same as that underlying the ME architecture. We simply extend the probability model to allow nested sequences of labels to be chosen, corresponding to the nested sequence of regions needed to specify a leaf of the tree. The probability model for a two-level tree is as follows:

$$P(\mathbf{y}|\mathbf{x}, \mathbf{\Theta}) = \sum_i P(i|\mathbf{x}, \boldsymbol{\eta}) \sum_j P(j|i, \mathbf{x}, \boldsymbol{\nu}_i) P(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_{ij}), \tag{10}$$

which corresponds to a choice of label i with probability $P(i|\mathbf{x}, \boldsymbol{\eta})$, followed by a conditional choice of label j with probability $P(j|i, \mathbf{x}, \boldsymbol{\nu}_i)$. This probability model yields the following log likelihood function:

$$l(\mathcal{X}, \mathbf{\Theta}) = \sum_l \log \sum_i P(i|\mathbf{x}^{(l)}, \boldsymbol{\eta}) \sum_j P(j|i, \mathbf{x}^{(l)}, \boldsymbol{\nu}_i) P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_{ij}), \tag{11}$$

where as in the one-level case the prior probabilities $g_i^{(l)} = P(i|\mathbf{x}^{(l)}, \boldsymbol{\eta})$ and $g_{j|i}^{(l)} = P(j|i, \mathbf{x}^{(l)}, \boldsymbol{\nu}_i)$ are defined in terms of underlying variables ξ_i and ξ_{ij} using the softmax function (cf. Equation 2). We also use Bayes' rule to define posterior probabilities in the obvious way:

$$h_i^{(l)} = \frac{g_i^{(l)} \sum_j g_{j|i}^{(l)} P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_{ij})}{\sum_j g_j^{(l)} \sum_k g_{k|j}^{(l)} P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_{jk})} \tag{12}$$

and

$$h_{j|i}^{(l)} = \frac{g_{j|i}^{(l)} P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_{ij})}{\sum_j g_{j|i}^{(l)} P(\mathbf{y}^{(l)}|\mathbf{x}^{(l)}, \boldsymbol{\theta}_{ij})}. \tag{13}$$

The posterior probability $h_i^{(l)}$ can be viewed as the credit assigned to the i^{th} nonterminal in the tree, and the posterior probability $h_{j|i}^{(l)}$ is the credit assigned to the branches below the nonterminals. The product $h_i^{(l)} h_{j|i}^{(l)}$ is therefore the credit assigned to expert (i, j).

A recursive relationship is available to compute the posterior probabilities efficiently in deep trees. The recursion proceeds upward in the tree, passing the denominator of the conditional posterior upward, multiplying by the priors, and normalizing (cf. the computation of h_i from $h_{j|i}$ in Equations 12 and 13).

We obtain a gradient ascent learning algorithm by computing the partial derivatives of l. If, as in the previous section, we assume linear experts and a linear gating network, as well as Gaussian probabilities for the experts, we obtain the following LMS-like learning algorithm:

$$\Delta \boldsymbol{\theta}_{ij} = \rho h_i^{(l)} h_{j|i}^{(l)} (\mathbf{y}^{(l)} - \boldsymbol{\mu}_{ij}^{(l)}) \mathbf{x}^{(l)^T}, \tag{14}$$

$$\Delta \boldsymbol{\eta}_i = \rho (h_i^{(l)} - g_i^{(l)}) \mathbf{x}^{(l)^T}, \tag{15}$$

and

$$\Delta \boldsymbol{\nu}_{ij} = \rho h_i^{(l)} (h_{j|i}^{(l)} - g_{j|i}^{(l)}) \mathbf{x}^{(l)^T}. \tag{16}$$

Each of these partial derivatives have a natural interpretation in terms of credit assignment. Credit is assigned to an expert by taking the product of the posterior probabilities along the path from the root of the tree to the expert (cf. Equation 14). The updates for the gating networks move the prior probabilities at a nonterminal toward the corresponding posterior probabilities, weighting these updates by the product of the posterior probabilities along the path from the root of the tree to the nonterminal in question (cf. Equation 16).

4 An EM Algorithm

Jordan and Jacobs (1994) have derived an Expectation-Maximization (EM) algorithm for estimating the parameters of the ME and HME architectures. This algorithm, an alternative to gradient methods, is particularly useful for models in which the expert networks and gating networks have simple parametric forms. Each iteration of the algorithm consists of two phases: (1) a recursive propagation upward and downward in the tree to compute posterior probabilities (the "E step"), and (2) solution of a set of local weighted maximum likelihood problems at the nonterminals and terminals of the tree (the "M step"). Jordan and Jacobs (1994) tested this algorithm on a nonlinear system identification problem (the forward dynamics of a four-degree-of-freedom robot arm), and report that it converges rapidly, converging nearly two orders of magnitude faster than backpropagation in a comparable multilayer perceptron network.

5 Discussion

Supervised learning can be viewed as the problem of approximating a nonlinear function from noisy samples of the function. Function approximation in neural networks has traditionally been based on superpositions of simple basis functions (typically either logistic functions or gaussians). The work described in this article can be viewed as an alternative approach to function approximation that is based on the principle of divide-and-conquer rather than the principle of superposition. It is important to emphasize that neither divide-and-conquer or superimposition is likely to work best in all cases; rather, the goal should be to characterize those classes of problems for which the different approaches are most appropriate. The use of probabilistic models can be helpful in this regard, clarifying the implicit assumptions made by the different forms of function approximator. Finally, in many problems the issue of learning speed is as important as the issue of approximation accuracy. In the area of learning speed divide-and-conquer approaches appear to have a natural advantage.

We conclude with a brief list of pointers to additional papers. The problem of model selection for HME architectures has been addressed by a number of authors. Several papers propose the use of greedy search procedures combined with some form of penalization for model complexity (Fritsch, Finke, and Waibel, 1997; Ramamurti and Ghosh, 1996; Saito and Nakano, 1996; Waterhouse and Robinson, 1994). Bayesian approaches for model selection or model averaging have also been presented, including methods based on Gibbs sampling (Jacobs, Peng, and Tanner, 1997) and variational approximation (Waterhouse, MacKay, and Robinson, 1994).

Theoretical analyses of the approximation and estimation rates for the ME (Zeevi, et al., 1998) and the HME (Jiang & Tanner, 1998) are available; the basic result is that the HME achieves an approximation rate of $O(m^{-2/s})$, where m is the number of experts and s is the dimensionality of the input vector. Jordan and Xu (1995) present an analysis of the convergence rate for the EM algorithm for the HME. Kang and Oh (1997) provide an analysis of the HME using tools from statistical physics; in particular, they show that successive partitions in the HME can be analyzed as phase transitions.

There is a broad literature on engineering applications of the HME architecture, including applications to state-space filtering, optimization, control, vision, speech recognition, speaker identification and time series analysis. Biological applications of the ME architecture have been presented by Jacobs and Kosslyn (1994) and Nowlan and Sejnowski (1993).

Acknowledgments

Preparation of this paper was supported in part by a grant from ATR Human Information Processing Laboratories.

References

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

Fritsch, J., Finke, M., and Waibel, A., 1997, Adaptively growing hierarchical mixtures of experts, in *Advances in Neural Information Processing Systems 9*, (M. Mozer, M. Jordan, and T. Petsche, Eds.), Cambridge, MA: MIT Press.

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

Jacobs, R. A., Peng, F., and Tanner, M. A., 1997, A Bayesian approach to model selection in hierarchical mixtures-of-experts architectures, *Neural Networks, 10*, 231–241.

Jacobs, R. A., and Kosslyn, S. M., 1994, Encoding shape and spatial relations: The role of receptive field size in coordinating complementary representations. *Cognitive Science, 18*, 361–386.

Jiang, W., and Tanner, M. T., 1998, Hierarchical mixtures-of-experts for exponential family regression models with generalized linear mean functions: Approximation and maximum likelihood estimation, in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, (G. F. Cooper and S. Moral, Eds.), San Francisco: Morgan Kaufmann.

* Jordan, M. I. & Jacobs, R. A., 1994, Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, *6*, 181–214.

Jordan, M. I., and Xu, L., 1995, Convergence properties of the EM approach to learning in mixture-of-experts architectures. *Neural Networks*, *8*, 1409–1431.

Kang, K., Oh, J-H., 1997, Statistical mechanics of the mixture of experts, in *Advances in Neural Information Processing Systems 9*, (M. Mozer, M. Jordan, and T. Petsche, Eds.), Cambridge, MA: MIT Press.

Nowlan, S. J., and Sejnowski, T., 1993, Filter selection model for generating visual motion signals, in *Advances in Neural Information Processing Systems 5*, (S. J. Hanson, J. D. Cowan, and C. L. Giles, Eds.), San Mateo, CA: Morgan Kaufmann.

Ramamurti, V., and Ghosh, J., 1996, Structural adaptation in mixture of experts, in *Proceedings of the 13th International Conference on Pattern Recognition*. Los Alamitos, CA: IEEE Computer Society Press.

Saito, K., and Nakano, R., 1996, A constructive learning algorithm for an HME, in *Proceedings of the IEEE International Conference on Neural Networks*.

Waterhouse, S. R., and Robinson, A. J., 1994, Constructive algorithms for hierarchical mixtures of experts, in *Advances in Neural Information Processing Systems 8*. (Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E., Eds.), Cambridge, MA: MIT Press.

Waterhouse, S., MacKay, D., and Robinson, T., 1994, Bayesian methods for mixtures of experts, in *Advances in Neural Information Processing Systems 8*. (Touretzky, D.S., Mozer, M.C., and Hasselmo, M.E., Eds.), Cambridge, MA: MIT Press.

Zeevi, A., Meir, R., and Maiorov, V., 1998, Error bounds for functional approximation and estimation using mixtures of experts. *IEEE Transactions on Information Theory, 44*, 1010–1025.