GAMLS: A Generalized framework for Associative Modular Learning Systems

Shailesh Kumar and Joydeep Ghosh

Department of Electrical and Computer Engineering, UT Austin, Austin TX 78712, USA

ABSTRACT

Learning a large number of simple local concepts is both faster and easier than learning a single global concept. Inspired by this principle of divide and conquer, a number of modular learning approaches have been proposed by the computational intelligence community. In modular learning, the classification/regression/clustering problem is first decomposed into a number of simpler subproblems, a module is learned for each of these subproblems, and finally their results are integrated by a suitable combining method. Mixtures of experts and clustering are two of the techniques that are describable in this paradigm.

In this paper we present a broad framework for Generalized Associative Modular Learning Systems (GAMLS). Modularity is introduced through soft association of each training pattern with every module. The coupled problems of learning the module parameters and learning associations are solved iteratively using deterministic annealing. Starting at a high temperature with only one module, GAMLS framework automatically evolves the required number of modules through a systematic growing and pruning technique. Each phase begins by splitting every module in the previous phase into two, updating these new modules and then pruning and merging any redundant modules. A phase transition is induced by temperature decay.

A number of existing modular learning problems, both unsupervised (clustering, mixture model density, mixture of principal components) and supervised (mixture of experts, radial basis function networks), can be effectively tackled in GAMLS. Case studies for clustering and regression using mixture of experts are provided for a number of datasets showing the efficacy of the GAMLS framework in evolving the right number of modules, inducing interpretable localizations among modules and robustness of the solution obtained. More importantly, this framework provides a unifying view for understanding and characterizing modular learning methods.

Keywords: Modular Learning, Deterministic Annealing, Model Selection, Associative Learning

1. INTRODUCTION

1.1. The learning problem

Learning to predict from data is a central problem in pattern recognition. Two types of predictive learning problems are the classification problem and the function approximation or regression problem.[1,2] In classification, the predicted value is a discrete class label $\omega \in \Omega = \{\omega_1, \omega_2, ..., \omega_C\}$ and in regression, the predicted value $y \in \mathcal{R}$ is a real number. In both cases the input data is a vector $\mathbf{x} \in \mathcal{R}^d$ in a d dimensional Euclidean space. In parametric supervised learning, the predictor $\phi(\mathbf{x}, \lambda)$ is assumed to have certain parametric form, e.g. multilayer perceptron (MLP), radial basis function network (RBF), mixture of experts (MOE) etc., with parameters $\lambda \in \mathcal{R}^P$. These parameters are estimated or "learned" using a labelled training set $\mathcal{X} = \{(\mathbf{x}^{(n)}, t^{(n)})\}_{n=1}^N$, where $t^{(n)}$ is the target output corresponding to the n^{th} input vector $\mathbf{x}^{(n)}$. Mean squared error of prediction or log likelihood criterion are generally used as objective functions and stochastic hill climbing techniques are typically used to obtain a reasonable set of parameters. In contast, unsupervised learning places more emphasis on finding structure in the data rather than on prediction. Three common unsupervised learning problems are clustering, probability density modelling and principal component analysis.

Further author information: (Send correspondence to Joydeep Ghosh)
Joydeep Ghosh: E-mail: ghosh@ece.utexas.edu
Shailesh Kumar: E-mail: skumar@cs.utexas.edu

1.2. Modular Learninig

Inspired by divide and conquer strategy, modular learning has been found to learn concepts more effectively (better performance) and more efficiently (faster learning) because *learning a large number of simple local concepts is both easier and more useful than learning a single complex global concept.* Modular learning, therefore, is a two stage process wherein the first stage deals with a useful decomposition of the problem into simpler subproblems, and the second stage deals with solving the subproblems and finding a means of combining the solutions of these subproblems to yield the final solution. Advantages of problem decomposition include the ease and efficiency in learning, scalability, interpretability, transparency etc.[3–7] The first isue of finding a useful problem decomposition has been addressed by a number of researchers.[6–11] Some of the useful ways of problem decomposition are based on (1) *functional modularity* of the system where different functional aspects are modeled independently and the complete system functionality is modeled by a combination of these individual functional models,[11] (3) *decomposition of the domain* of the model by specializing different models in different parts of the input space (e.g. different control laws apply in different local environments, learning mixture of experts, mixture models for classification and regression tasks etc.)[8] (4) *goal decomposition* where the overall goal is decomposed into a number of subgoals each of which is pursued by a simple model and the overall goal falls out as a result of the combination of these subgoals (e.g. pairwise classification),[12] and finally (5) *state decomposition* where different models are learnt to deal with different states in which the system can be at any time (e.g. time series applications).[7]

The second issue of combining the solutions of the individual subproblems can again be addressed in a number of ways. A general approach is to look at the output of the overall system as a linear combination of outputs of individual local models.[13–15] Different constraints and interpretations of the weighing functions that form the coefficients of the linear combination lead to different approaches to the combining problem. A probabilistic interpretation with a constraint that all weighting coefficients are non-negative and add to one leads to a approaches like Belief Networks[16] and uncertain reasoning,[17] and is used in supervised learning of predictors like Mixture of Experts, Mixture Model, Mixture of Gaussian, Radial Basis functions, Bayesian learning[18,19] etc. Fuzzy class membership interpretation to the weighing coefficients lead to methods like fuzzy clustering[20] fuzzy pattern classifiers[21] and fuzzy controls etc.[22] Simple weighted averaging approaches to combining lead to local learning algorithms,[23] local linear regression,[24] weighted nearest neighbor,[25] etc. These approaches to combining involve a *soft decomposition* of the problem, whereby the transition from choosing the output from one model to another is smooth. Another approach, used mostly in symbolic domains, is *hard decomposition* where the weighing coefficients are discrete indicator functions that pick exactly one of the local models at any time. The weighing coefficients will take only binary values 0 or 1 (as against real numbers in soft decomposition). Common examples of this approach are classification and regression trees (CART)[26] and k-nearest neighbors.

Another issue implicit in modular learning is to find an appropriate number of modules, which depends on the complexity of the problem and the form of the modules used. For example, in clustering, the number of clusters, or in mixture of experts, the number of mixtures required is not known just from the data. Various techniques have been used for different problems to determine the number of modules to be used.[27–29]

In this article, a framework for generalized associative modular learning systems (GAMLS) for both supervised and unspuervised paradigms is proposed. In this framework, localization is induced by association of training patterns with different modules. The association between data points and the modules, as well as all the module parameters are updated as the learning progresses. This framework is applied to the mixture of experts and clustering for illustration. A growing and prunning strategy for determining the appropriate number of modules, independent of the specific learning problem, is also incorporated in the GAMLS framework.

2. GENERALIZED ASSOCIATIVE MODULAR LEARNING FRAMEWORK

In the GAMLS framework, each module, indexed m, is associated with a set of module parameters λ_m and optionally, a set of kernel parameters θ_m (only for supervised learning prediction tasks). The goal in GAMLS is to learn $\Lambda = \{\lambda_m\}_{m=1}^{M}$, $\Theta = \{\theta_m\}_{m=1}^{M}$ (if present) and to find an appropriate number of modules M from the training data $\mathcal{X}$ such that an objective is optimized. This involves four coupled steps:

1. *Model Selection* or finding an appropriate number and form ‘of modules to learn the concept. In GAMLS deterministic annealing is used to evolve modules (and kernels) and at a given scale[30] (inverse temperature), the number of modules is fixed. With every phase transition[31] (change in temperature), the number of modules, M, is updated through a systematic growing and prunning mechanism.

2. *Modularization* of the problem among the M parametric modules such that each one specializes on some local aspect of the problem. In the GAMLS framework, modularization is done through soft association of data points with different modules, using deterministic annealing approach. Modularization is required both in supervised and unsupervised learning tasks.

3. *Specialization* of each module through training over the relevant input data. In GAMLS, a hill climbing approach is used to optimize an objective function. Specialization is required in both supervised and unsupervised learning tasks.

4. *Combination* of the outputs of these specializing modules to yield a final output, using (1). Combination is required only in supervised learning tasks.

2.1. Modularization by Association

In the GAMLS framework, each module is associated with a (soft) subset of the training data. This approach automatically brings about modularization with respect to the input or the output space and at the same time is not restricted to these two.

Let $0 \leq \alpha_m^{(n)} \leq 1$ be the *association* between the n^{th} data point in the training set and the m^{th} module. We define an $N \times M$ association matrix $\mathbf{A} = [\alpha_m^{(n)}]$ where each row corresponds to one data point and each column corresponds to one module. These associations satisfy the completeness constraint namely, $\sum_{m=1}^{M} \alpha_m^{(n)} = 1, \forall n = 1...N$. These associations are used to obtain the the overall output:

$$\Phi(\mathbf{x}, \Lambda, \Theta) = \sum_{m=1}^{M} \alpha_m(\mathbf{x}, \Theta)\phi_m(\mathbf{x}, \lambda_m). \tag{1}$$

In supervised learning the association of a novel data point $\mathbf{x}$, with the m^{th} module, $\alpha_m(\mathbf{x}, \Theta)$ is computed using normalized kernel function outputs:

$$\alpha_m(\mathbf{x}, \Theta) = \frac{\mathcal{K}_m(\mathbf{x}, \theta_m)}{\sum_{j=1}^{M} \mathcal{K}_j(\mathbf{x}, \theta_j)}. \tag{2}$$

Decoupling between different modules is measured in terms of the average entropy of the association matrix, $\mathcal{H}(\mathbf{A}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{m=1}^{M} \alpha_m^{(n)} \log(\alpha_m^{(n)})$ A high entropy value indicates a high degree of coupling or sharing of data points among the different modules. It is desired that as the learning progresses, $\mathcal{H}(\mathbf{A})$ should decrease.

2.2. Module Specialization

In GAMLS framework, the modules (and kernels) are updated based on an appropriate objective function and using the dataset associated with it.

2.2.1. Supervised Learning

In supervised learning, the goal is to minimize the disagreement between the target output t and the output of the predictor for the input $\mathbf{x}$ associated with the target t. Two learning paradigms, namely *global learning* and *local learning*, can be used to model this disagreement.[6] In *global learning* the disagreement between the target output $t^{(n)}$ and the overall predictor output $\Phi(\mathbf{x}^{(n)}, \Lambda, \mathcal{A}^{(n)})$ is used. Here $\mathcal{A}^{(n)} = \{\alpha_m^{(n)}\}_{m=1}^{M}$ is the n^{th} row of the association matrix $\mathbf{A}$. Thus

$$J_g(\Lambda, \mathbf{A}, \mathcal{X}) = \sum_{n=1}^{N} \delta\left(t^{(n)}, \Phi(\mathbf{x}^{(n)}, \Lambda, \mathcal{A}^{(n)})\right), \tag{3}$$

where $\delta(a, b)$ is a measure of disagreement between a and b (e.g. squared Euclidean distance).

The hill climbing update rule for the k^{th} module parameters λ_k for global learning is given by:

$$\Delta\lambda_k \propto \sum_{n=1}^{N} \alpha_k^{(n)} \left(\frac{\partial\delta\left(t^{(n)}, \Phi(\mathbf{x}^{(n)}, \Lambda, \mathcal{A}^{(n)})\right)}{\partial\Phi(\mathbf{x}^{(n)}, \Lambda, \mathcal{A}^{(n)})}\right) \left(\frac{\partial\phi_k(\mathbf{x}^{(n)}, \lambda_k)}{\partial\lambda_k}\right). \tag{4}$$

This shows that the update in λ_k depends on the current values of all the parameters Λ and not just the current value of λ_k. Thus there is significant coupling among the parameters of different modules. Moreover, the parameters of any module cannot be interpreted independently of the other modules, which means they cannot be seen as local approximations of the underlying system.[6]

Local learning is an alternative to global learning that is less prone to these disadvantages. The (local) objective function is:

$$J_l(\Lambda, \mathbf{A}, \mathcal{X}) = \sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_m^{(n)} \delta\left(t^{(n)}, \phi_m(\mathbf{x}^{(n)}, \lambda_m)\right). \tag{5}$$

The update rule for module parameter λ_k in local learning is given by:

$$\Delta\lambda_k \propto \sum_{n=1}^{N} \alpha_k^{(n)} \left(\frac{\partial \delta(t^{(n)}, \phi_k(\mathbf{x}^{(n)}, \lambda_k))}{\partial \phi_k(\mathbf{x}^{(n)}, \lambda_k)}\right) \left(\frac{\partial \phi_k(\mathbf{x}^{(n)}, \lambda_k)}{\partial \lambda_k}\right). \tag{6}$$

Thus the update of λ_k depends only indirectly on other module's parameters (via $\mathbf{A}$). Comparing equation (4) and (6) it is obvious that local learning is far less prone to coupling among parameters of different modules than global learning. The GAMLS framework, therefore, uses the local learning paradigm for module specialization. Detailed comparison between global and local learning is done by Murray-Smith and Johansen.[32]

2.2.2. Unsupervised Learning

In unsupervised learning there is no target value to be matched. Instead, the objective function is based on the nature of the structure sought in the data. Let $\tilde{\delta}(\mathbf{x}^{(n)}, \lambda_m)$, denote the cost of associating the n^{th} data point with m^{th} module. The overall objective function $\tilde{J}_l$ for unsupervised local learning is:

$$\tilde{J}_l(\Lambda, \mathbf{A}, \mathcal{X}) = \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha_m^{(n)} \tilde{\delta}(\mathbf{x}^{(n)}, \lambda_m). \tag{7}$$

The update for the parameters λ_k is given by:

$$\Delta\lambda_k \propto \sum_{n=1}^{N} \alpha_m^{(n)} \frac{\partial \tilde{\delta}(\mathbf{x}^{(n)}, \lambda_m)}{\partial \lambda_k}. \tag{8}$$

The updates in equations (6) and (8) can be solved analytically by setting $\Delta\lambda_k = 0$ or by hill climbing using a user defined learning rate η.

2.3. Association Update

Association update is done alternately with module (and kernel) parameter update. Again there is a distinction between supervised and unsupervised learning tasks for association update.

2.3.1. Supervised Learning

In supervised learning, the kernel functions provide for localization bias to each module. In this work we consider the input localization, however other kernel functions can be used for other types of localizations too. The association $\alpha_m^{(n)}$ is updated in such a way that it is high if the cost $\delta(t^{(n)}, \phi_m(\mathbf{x}^{(n)}, \lambda_m))$ is low and the kernel output $\mathcal{K}_m(\mathbf{x}, \theta_m)$ is high:

$$\alpha_m^{(n)} \leftarrow \frac{\mathcal{K}_m(\mathbf{x}^{(n)}, \theta_m) \exp\left\{-\beta\delta(t^{(n)}, \phi_m(\mathbf{x}^{(n)}, \lambda_m))\right\}}{\sum_{j=1}^{M} \mathcal{K}_j(\mathbf{x}^{(n)}, \theta_j) \exp\left\{-\beta\delta(t^{(n)}, \phi_j(\mathbf{x}^{(n)}, \lambda_j))\right\}}, \tag{9}$$

where β is the inverse of the temperature parameter and denominator is used to normalize the association values so that they satisfy the completeness constraint.

2.3.2. Unsupervised Learning

In unsupervised learning there are no kernel functions. Hence the association update is obtained directly from the Gibbs distribution that minimizes (7) such that the association $\alpha_m^{(n)}$ is high if the cost $\tilde{\delta}(\mathbf{x}^{(n)}, \lambda_m)$ is low:

$$\alpha_m^{(n)} \leftarrow \frac{\exp\left\{-\beta\tilde{\delta}(\mathbf{x}^{(n)}, \lambda_m)\right\}}{\sum_{j=1}^{M}\exp\left\{-\beta\tilde{\delta}(\mathbf{x}^{(n)}, \lambda_j)\right\}}. \tag{10}$$

2.4. Kernel Update

In supervised learning the kernel parameters Θ are also updated to refine the localization of each module. The idea is to increase the kernel output for those data points whose association is high. Hence the following objective is used for Θ update:

$$J_{\mathcal{K}}(\Theta, \mathcal{X}, \mathbf{A}) = \sum_{n=1}^{N}\sum_{m=1}^{M}\alpha_m^{(n)}\mathcal{K}_m(\mathbf{x}^{(n)}, \theta_m). \tag{11}$$

And the update rule for the kernel parameter θ_m is given by:

$$\Delta\theta_k \propto \sum_{n=1}^{N}\alpha_k^{(n)}\frac{\partial\mathcal{K}_k(\mathbf{x}^{(n)}, \theta_k)}{\partial\theta_k}. \tag{12}$$

2.5. Growing and Pruning Algorithm

‘ As mentioned above, deterministic annealing is used to evolve the modules (and kernels) in the GAMLS framework. The learning process starts at a high temperature (low β) with only $M = 1$ module (and 1 kernel). All data points have an association 1 with this module. Once the module (and kernel) parameters are learned, the temperature is reduced and the module is split into two by small random perturbations in the parameters. The new modules (and kernels) are again learned at the lower temperature till the percentage decrease in the cost J_l is less than a user specified threshold. The learned modules are pruned and merged if necessary and the process of splitting is continued till the temperature has cooled down sufficiently. The complete algorithm is described below:

1. *Initialize* scale parameter $\beta = \beta_0$ (some low value), $M = 1$ and λ_1, θ_1 based on the data set.
2. $\beta \leftarrow \beta\Delta\beta$, ($\Delta\beta \geq 1$ is a user defined scale growth or temperature decay factor).
3. $\forall m = 1, ..., M$
   - $\lambda_{2m-1} \leftarrow$ PERTURB(λ_m), $\lambda_{2m} \leftarrow$ PERTURB(λ_m).
   - (supervised) $\theta_{2m-1} \leftarrow$ PERTURB(θ_m), $\theta_{2m} \leftarrow$ PERTURB(θ_m).
4. Update Associations $\alpha_m^{(n)}$ (m=1...2M, n=1...N) using (9) (supervised) or (10) (unsupervised).
5. Update λ_m (m=1...2M), and, for supervised learning, also θ_m.
6. If $\frac{\Delta J_l}{J_l} < \gamma$ (supervised) or $\frac{\Delta\tilde{J}_l}{\tilde{J}_l} < \gamma$ (unsupervised) (where γ is a user defined threshold on percentage change in cost) then go to step (7) otherwise go to step (4).
7. (Λ, Θ, M) = PRUNEandMERGE($\mathbf{A}$, Λ, Θ)*
8. If $\beta < \beta_f$, then go to step 2 otherwise return Λ, Θ.

*(see PRUNEandMERGE algorithm below; Θ only for supervised learning.)

2.6. Prune and Merge Algorithm

At step 7 of the above algorithm, at any scale β, the number of modules is obtained by first doubling the number of modules at the previous temperature, learning these modules (and kernel) parameters and then pruning to eliminate redundant modules. Note that the following algorithm does not depend on the module or kernel parameters, it is very generic and independent of the application. The PRUNEandMERGE($\mathbf{A}$, Λ, Θ) algorithm is as follows:

1. *PRUNE*: $\forall m = 1...M$, if $w_m = \sum_{n=1}^{N} \alpha_m^{(n)} < w_0$ (a user defined parameter) then prune module m, i.e. $\mathbf{A} = \mathbf{A} \sim \mathcal{A}_m$ (remove column $\mathcal{A}_m$) from $\mathbf{A}$ and renormalize $\mathbf{A}$ so that all rows add up to 1.
2. *MERGE*: $\forall$ column pairs $(i \neq j)$, if *association distance* $\mathcal{D}(\mathcal{A}_i, \mathcal{A}_j) < d_0$ (a user defined parameter) then merge columns i and j i.e. (i) add column $\mathcal{A}_{ij} = \mathcal{A}_i + \mathcal{A}_j$ to $\mathbf{A}$ and (ii) delete columns $\mathcal{A}_i$ and $\mathcal{A}_j$ from $\mathbf{A}$. Repeat this till there is no more merge required. Two possible distance functions are the L_p norm distance of the mean KL distance.
3. Now using the resulting association matrix, update all the module and kernel parameters as in steps 5.
4. Return the updated $\mathbf{A}$, Λ and Θ.

2.7. Association Generalization

For supervised learning problems, after all the modules and kernels are learned using the above algorithm, then for a new input point $\mathbf{x}$, the overall output $\Phi(\mathbf{x}, \Lambda, \Theta)$ is given by equation (1) where the association functions $\alpha_m(\mathbf{x}, \Theta)$ are given by equation (2). Hence the kernel functions are used to generalize the associations of a novel input point with different modules.

3. CASE STUDY: CLUSTERING

3.1. The Clustering Problem

The clustering problem deals with finding a set of clusters denoted by cluster centers $\Psi = \{\psi_m \in \mathcal{R}^d\}_{m=1}^{M}$ for a given dataset $\mathcal{X} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ such that the total distortion measure, given by $J(\Psi, \mathcal{X}) = \sum_{n=1}^{N} \min_{m=1..M} ||\mathbf{x}^{(n)} - \psi_m||^2$, is minimum. Various authors have looked at clustering using deterministic annealing[31,33] and scale based approaches.[30] The GAMLS framework leads to the same formulation for the clustering problem.

3.2. GAMLS for clustering

The clustering problem fits very well in the GAMLS framework by considering each cluster as a module with parameter $\lambda_m \equiv \psi_m$, the mean of the m^{th} cluster. For any number M of clusters (at some scale β), define an $N \times M$ association matrix $\mathbf{A} = [\alpha_m^{(n)}]$ and cost $\tilde{\delta}(\mathbf{x}^{(n)}, \psi_m) = ||\mathbf{x}^{(n)} - \psi_m||^2$. Hence the local learning objective function for clustering using GAMLS formulation (equation 7) is given by:

$$\tilde{J}_l(\Psi, \mathcal{X}) = \sum_{n=1}^{N} \sum_{m=1}^{M} \alpha_m^{(n)} ||\mathbf{x}^{(n)} - \psi_m||^2. \tag{13}$$

From equation (8) the analytical form for mean update is given by:

$$\psi_k = \frac{\sum_{n=1}^{N} \alpha_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \alpha_k^{(n)}}. \tag{14}$$

The association update from equation (10) is given by:

$$\alpha_m^{(n)} \leftarrow \frac{\exp\left\{-\beta ||\mathbf{x}^{(n)} - \psi_m||^2\right\}}{\sum_{j=1}^{M} \exp\left\{-\beta ||\mathbf{x}^{(n)} - \psi_j||^2\right\}}. \tag{15}$$

The clustering algorithm starts with initializing just 1 cluster for the whole data set and setting the mean of this cluster as the mean of the entire dataset. Subsequent clusters are evolved using the algorithm given in section 2.5 and using the appropriate update equations given above.

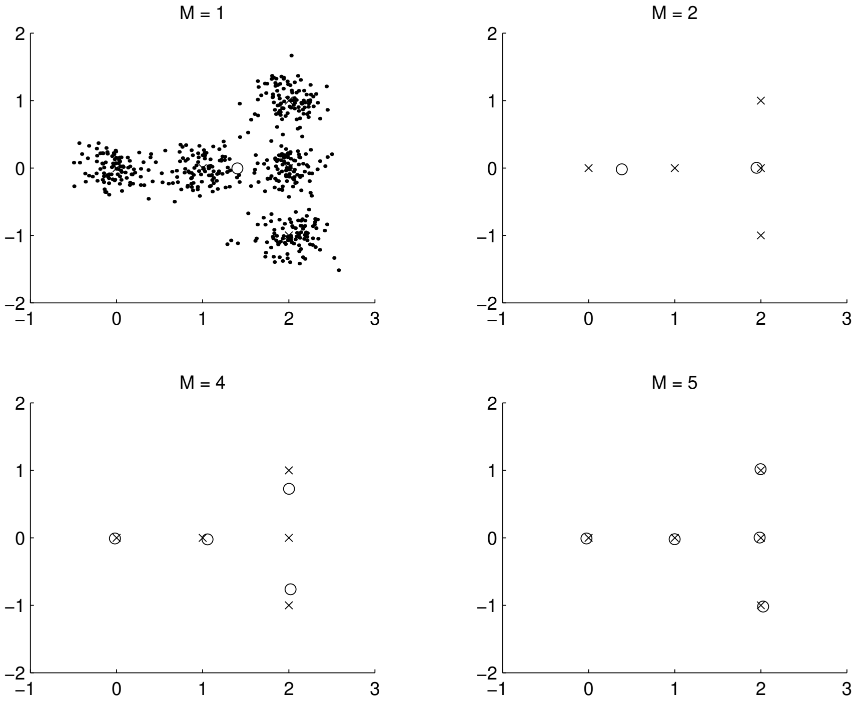

Figure 1. **Clustering Result**: $\times$ denotes the actual cluster centers from which the data was generated. $\circ$ denotes the cluster centers learned by the GAMLS clustering algorithm at different scales.

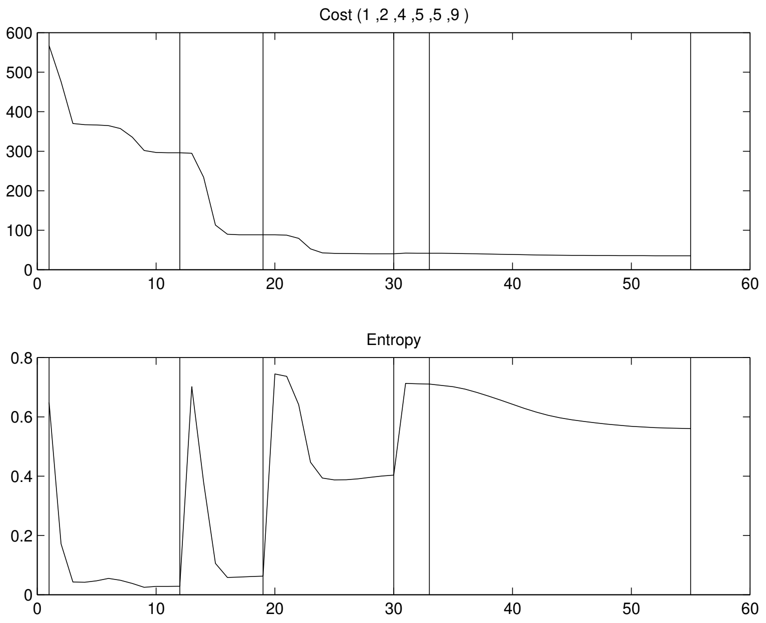

Figure 2. **Phase transitions**: The numbers in the *Cost* (top) subplot title denote the number of clusters (M) in each phase. The vertical lines denote the phase transitions. The horizontal axis is the iteration count.

3.3. Experiments and Results

The GAMLS framework was evaluated over a number of datasets. As compared to the k-means clustering, clustering using GAMLS was found to be more robust to cluster initializations. Since in GAMLS clustering, there is no random initialization, for a given set of learning parameters $(\beta_0, \Delta\beta, \gamma, d_0, w_0)$ the results were almost always the same. Figure 1 shows the result of applying clustering to a simple synthetic data shown in the top left plot. It was generated using five Gaussian with centers located at the points marked $\times$. The top left subplot shows the single cluster at the initial temperature of ($\frac{1}{\beta_0} = 0.1$) (marked $\circ$). The temperature was decreased by a factor of $\frac{1}{\Delta\beta} = 1.1$ and the initial cluster was split into two as shown in top right subplot marked by $\circ$. On further decreasing the temperature to 0.083 we get four clusters (bottom left subplot). At temperature 0.075, GAMLS clustering algorithm first creates 8 clusters and then prunes and merges three of them to give the right number of clusters, i.e. five. As shown by $\circ$ in the bottom right subplot of figure 1. These clusters overlap exactly with the cluster centers used for generating these clusters in the first place. On further decreasing the temperature 0.068, the number of clusters remain 5 while at the next lower temperature 0.062, the number of clusters become 9. The experiment was repeated a number of times with prune parameter $w_0 = \frac{1}{2M}$ (for the current number M of clusters) and merge distance threshold $d_0 = 0.2$. ($\mathcal{D}_e^{(2)}$ from equation (13)), $\gamma = 0.001$ (convergence threshold for percentage change in $\tilde{J}_l$ at any temperature). In all cases the results were exactly the same.

Figure 2 shows the cost function (7) and the average entropy of the association matrix $\mathcal{H}(\mathbf{A})$ against iteration count. The numbers above the top figure denote the number of clusters and the vertical lines in both the plots denote the phase transitions[31] i.e. where the convergence is reached at the current temperature (one cluster is at iteration count 0). The cost decreases continuously as the learning progresses across phase transitions. The entropy however shows a jump at the beginning of each phase and decreases subsequently. This is obvious because the two perturbed clusters generated at the beginning of each phase share the data points almost equally and as the learning progresses and these clusters are separated, the overlap between their data points decreases and so does the entropy of the association matrix.

The k-means clustering algorithm was also used on the same dataset for comparison and out of 1000 different trials, (each with a different random initialization for $k = 5$), it converged to local minima 11% of the time. Since in GAMLS framework the learning process does not start with random cluster centers, it is very robust and as suggested by the experiments, gives the same results for the same set of learning parameters while k-means clustering is very sensitive to cluster center initialization.

4. CASE STUDY: MIXTURE OF EXPERTS

4.1. GAMLS for Mixture of Experts

A mixture of experts[34] can be modeled in a GAMLS framework by choosing each expert as a module. The weights of the m^{th} expert $\mathbf{w}_m \equiv \lambda_m$ are the module parameters and $\Lambda \equiv \mathcal{W}$. The kernel functions of the GAMLS framework replace the gating network $\mathcal{G}$. Instead of using hyper-planes for the gating network, each gating network in this paper is denoted by a weighted Gaussian. Hence $\theta_m = \{\nu_m, \mu_m, Q_m\}$, $m = 1 \ldots M$, the prior of choosing kernel ν_m, mean μ_m and covariance Q_m of each of the m Gaussians. Hence, the output of the m^{th} kernel function is given by:

$$\mathcal{K}_m(\mathbf{x}, \nu_m, \mu_m, Q_m) = \frac{\nu_m}{(2\pi)^{\frac{d}{2}} |Q_m|^{\frac{1}{2}}} \exp\left[-0.5(\mathbf{x} - \mu_m)^{\dagger} Q_m^{-1} (\mathbf{x} - \mu_m)\right]. \tag{16}$$

While, this form of the kernel function is chosen for specific type of localization, the GAMLS framework does allow for different choices of the kernel function. The local learning objective function (based on equation 5) in terms of the association matrix, $\mathbf{A}$ is:

$$J_l(\mathcal{W}, \mathbf{A}, \mathcal{X}) = \sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_m^{(n)} \left(t^{(n)} - \mathbf{w}_m^{\dagger} \mathbf{x}^{(n)}\right)^2. \tag{17}$$

The update rule for weight parameters with fixed association values obtained from equation (6) is given by:

$$\mathbf{w}_m \leftarrow \left(\sum_{n=1}^{n} \alpha_m^{(n)} \left(\mathbf{x}^{(n)} \mathbf{x}^{(n)\dagger}\right)\right)^{-1} \left(\sum_{n=1}^{N} \alpha_m^{(n)} t^{(n)} \mathbf{x}^{(n)}\right). \tag{18}$$

The association matrix is updated using:

$$\alpha_m^{(n)} \leftarrow \frac{\mathcal{K}_m(\mathbf{x}, \nu_m, \mu_m, Q_m) \exp\left\{-\gamma \left(t^{(n)} - \mathbf{w}_m^{\dagger} \mathbf{x}^{(n)}\right)^2\right\}}{\sum_{j=1}^{M} \mathcal{K}_m(\mathbf{x}, \nu_m, \mu_m, Q_m) \exp\left\{-\gamma \left(t^{(n)} - \mathbf{w}_j^{\dagger} \mathbf{x}^{(n)}\right)^2\right\}}. \tag{19}$$

The update rule for the Gaussian parameters from equation (12) can be derived as:

$$\nu_m = \frac{1}{N}\sum_{n=1}^{N}\alpha_m^{(n)};\ \mu_m = \frac{\sum_{n=1}^{N}\alpha_m^{(n)}\mathbf{x}^{(n)}}{\sum_{n=1}^{N}\alpha_m^{(n)}};\ Q_m = \frac{\sum_{n=1}^{N}\alpha_m^{(n)}\left(\mathbf{x}^{(n)}-\mu_m\right)\left(\mathbf{x}^{(n)}-\mu_m\right)^{\dagger}}{\sum_{n=1}^{N}\alpha_m^{(n)}}. \tag{20}$$

4.2. Experiments and Results

A number of regression problems were used to compare the standard MOE with the GAMLS MOE described above. In all cases the GAMLS MOE performed significantly better in terms of prediction errors over test set, number of experts required, training time and interpretability of results. Figure 3 and 4 show the GAMLS MOE results for a univariate noiseless sinc function while figure 5 shows GAMLS MOE results for the function $y = \cos(6x^2)$+noise.[32] The top left subplot (in both figure 3 and 5) shows the training data. The top right subplot shows the output of the MOE after learning is complete. The GAMLS framework required 32 experts to learn the sinc function and 8 experts to learn the cosine function for the given set of parameters ($w_0 = \frac{1}{2M}, d_0 = 0.1, \gamma = 0.01$). Each of the experts is localized by a Gaussian kernel and the lines shown in the top left subplots represent the output of the dominant expert. The dots in the top right subplots show the actual GAMLS MOE output after learning is complete.

Figure 4 shows the dominant experts and the overall mixture output for different phases with M = 4, 8, 16 and 32. As expected, the ability of the mixture to model the sinc function increases with the number of linear experts allowed. However, in relatively linear regions like between $x \in [-1, 0]$ or $x \in [0, 1]$, there are more than required number of experts for $M = 32$. This is because the association distance between the experts in these regions is higher than the threshold d_0. Since different sets of data points are associated with different experts, the association distance is high. If instead, the distance between mixtures is measured directly in terms of the mixture parameters, then the redundant mixtures in the aforementioned regions would have merged. Hence this redundancy is an artifact of the localization method (using kernels) and the choice of the distance metric between two modules. The data is noiseless in this experiment hence the more experts the better, however when noise is present, as in the next example, the growing algorithm should be stopped earlier to prevent overfitting.

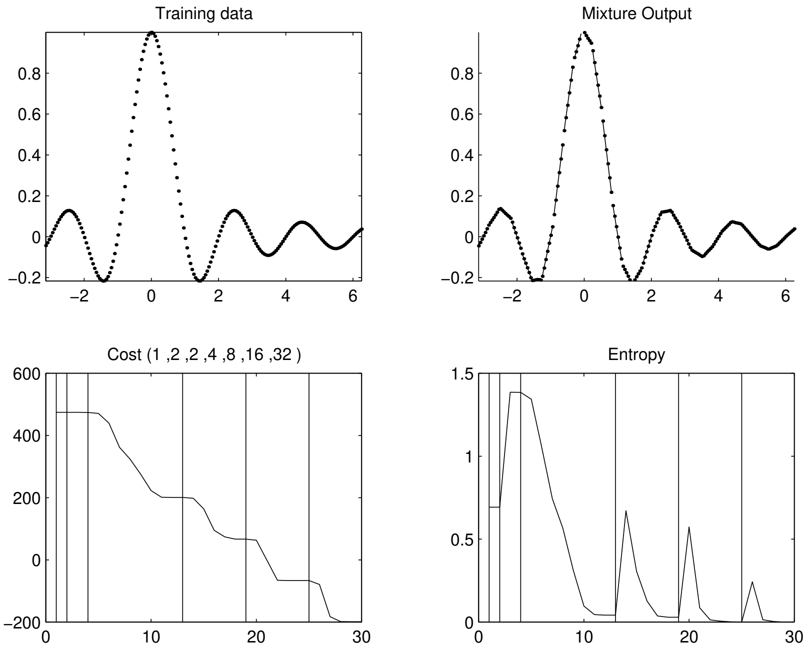

Figure 3. MOE regression for sinc function without noise: The number in the title of the bottom left (*Cost*) subplot indicate the number of experts in subsequent phases. The vertical lines in the bottom plots indicate phase transitions.

Results similar to those in figure 5 were obtained by Murray-Smith and Johansen[32] for this regression problem, however there are twho differences in their approach and GAMLS. First, the number of experts is not given to the GAMLS system, while in the local learning model used in,[32] a fixed number of experts were known apriori. GAMLS

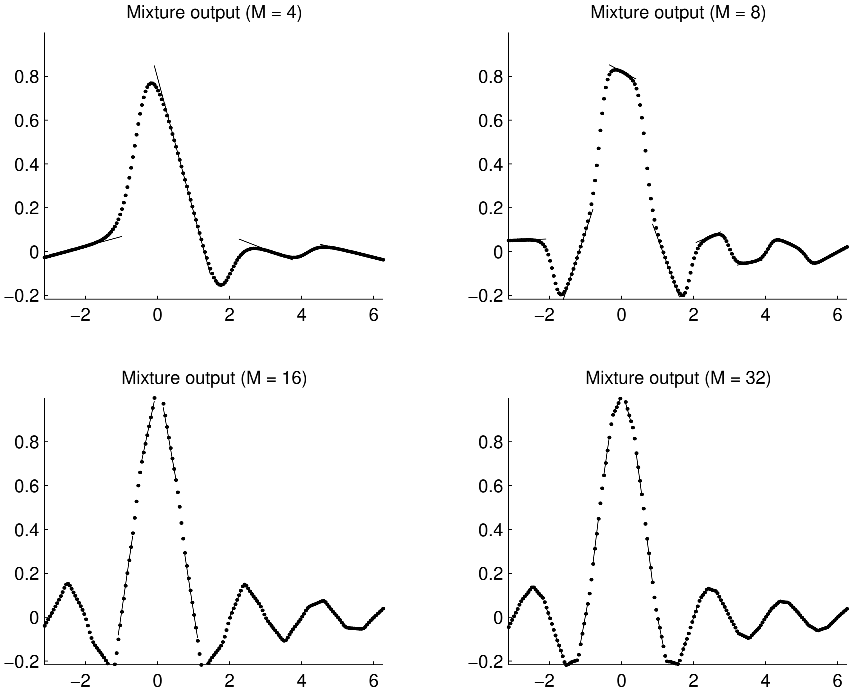

Figure 4. MOE regression for sinc function without noise for different M: The solid line shows the dominant expert output while the dots show the output of the complete mixture for different values of M.

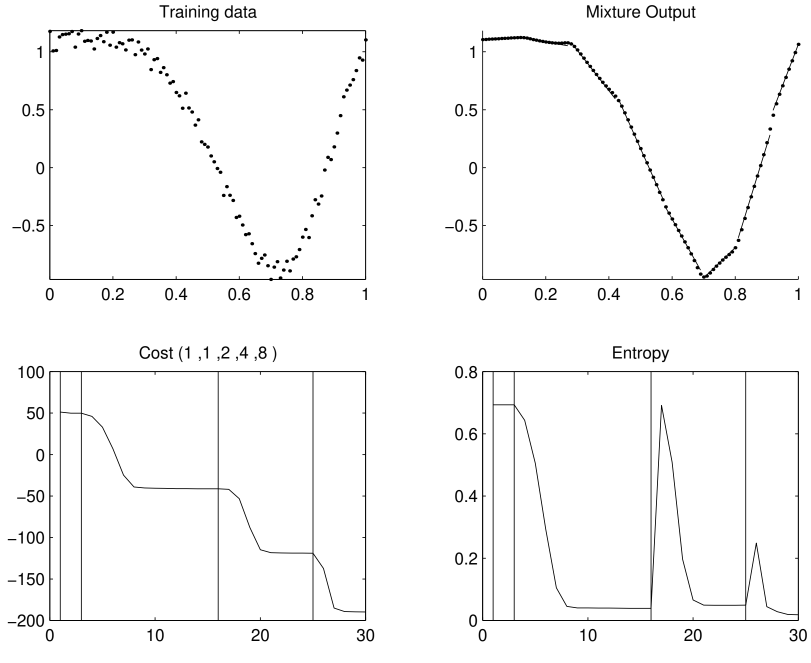

Figure 5. MOE regression for $\mathbf{y} = \mathbf{cos}(6x^2)$ function with $\sigma = 1$ noise

evolves the required eight experts, however, at lower temperatures more experts are added to the mixture and this leads to over fitting. Secondly, the kernels used in[32] were the radial basis functions which were initialized based only on the distribution of the input data, while in GAMLS the module parameters and kernel parameters are learned simultaneously and the kernel parameters are based on both the input distribution as well as the error in the output of different modules (via $\mathbf{A}$). One advantage of this approach can be seen in the top right subplot of figure 3. If the kernels were trained only based on the input distribution then they would have been equally distributed throughout the input space. However, it is reasonable to have more linear experts in the region with high curvature and less in the region with low curvature. In the GAMLS MOE shown in figure 3, this distribution of experts is very obvious.

The bottom left subplot in figure 3 and 5 show the cost function (5) as a function of iterations and the bottom right subplot shows the entropy of the association matrix. These plots are similar to the corresponding plots in figure 2 for clustering. At any temperature the error and entropy decreases till the stopping condition is reached. When at a lower temperature, the mixtures are reinitialized by random perturbations, the entropy shoots up again but drops down as learning of mixture and kernel parameters take place. The vertical lines in the bottom subplot depict the phase transition.[31] In both figures 3 and 5, the initial temperature was 1 and was decayed by a factor of 2 in each phase transition.

5. CONCLUSIONS

A generalized associative modular learning system was presented in this paper along with two case studies, one for the unsupervised clustering problem and another for the supervised mixture of experts based regression problems. The GAMLS framework uses ideas from deterministic annealing to evolve a modular learning system which adjusts the number of modules automatically as it learns and localizes each module. The task of problem decomposition was decoupled from module specialization through soft association of training data with different modules. The prune and merge algorithm, which uses only the association matrix, is very generic and is independent of the nature of modules. Additional flexibility to induce specialization in individual modules is provided by the kernel functions for supervised learning prediction problems.

In the clustering problem, GAMLS framework was particularly useful in finding the right number of clusters. It was also found to very more robust and gave the same results for the same set of learning parameters in all the experiments as against k-means clustering which was very sensitive to initialization even when the number of clusters is known beforehand. In the regression problem using mixture of experts, the GAMLS framework not only learned highly localized and interpretable experts, but also adjusted the number of experts required and their distribution in the input space. Further experiments suggested that GAMLS is tolerant to the various user defined input parameters. More work is required to analyze the effect of these parameters, their automatic selection based on the training data and overfitting issues in case of noisy data.

GAMLS provides a general framework for modular learning and provides for automatic adjustment of the number of modules required. It also fits very well into a number of modular learning frameworks like the mixture of experts, mixture of principal components, clustering and mixture of Gaussian. Finally it provides a unifying framework for the divide and conquer strategy for both unsupervised and supervised learning problem.

ACKNOWLEDGMENTS

This research was spported in part by ARO contracts DAAH04-94-G-0417 and 04-95-10494.

REFERENCES

1. C. M. Bishop, *Neural Networks for Pattern Recognition*, Oxford University Press, New York, 1995.
2. J. H. Friedman, "An overview of predictive learning and function approximation," in *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*, V. Cherkassky, J. Friedman, and H. Wechsler, eds., pp. 1–61, Springer Verlag, 1994.
3. D. Ballard, "Modular learning in neural networks," *Proc. AAAI-87* , pp. 279–284, 1987.
4. B. Happel and J. Murre, "Design and evolution of modular neural network architectures," *Neural Networks* **7:6/7**, pp. 985–1004, 1994.
5. V. Ramamurti and J. Ghosh, "Structurally adaptive modular networks for nonstationary environments," *IEEE Trans. on Neural Networks* **10**, pp. 152–160, Jan 1999.
6. R. Murray-Smith and T. A. Johansen, *Multiple Model Approaches to Modelling and Control*, Taylor and Francis, UK, 1997.
7. V. Petridis and A. Kehagias, *Predictive Modular Neural Networks: Applications to Time Series*, Kluwer Academic Publishers, Boston, 1998.

8. R. E. Jenkins and B. P. Yuhas, "A simplified neural network solution through problem decomposition: The case of the backer-upper," *IEEE Transactions on Neural Networks* **4**, pp. 718–720, July 1993.
9. B.-L. Lu and M. Ito, "Task decomposition and module combination based on class relations: A modular neural network for pattern classification," Tech. Rep. BMC TR-98-1, 1998.
10. C. G. Atkeson, A. W. Moore, and S. Schaal, "Locally weighted learning," *Artificial Intelligence Review* , 1996.
11. X. Zong-ben and K. Chung-ping, "A decomposition principle for complexity reduction of artificial neural networks," *Neural Networks* **9**(6), pp. 999–1016, 1996.
12. T. Hastie and R. Tibshirani, "Classification by pairwise coupling," in *Advances in Neural Information Processing Systems*, M. J. K. Michael I. Jordan and S. A. Solla, eds., vol. 10, pp. 507–513, (MIT Press, Cambridge Massachusetts), 1998.
13. S. Hashem, "Optimal linear combinations of neural networks," *Neural Networks* **10**(4), pp. 599–614, 1997.
14. M. Leblanc and R. Tibshirani, "Combining estimates in regression and classification," tech. rep., Department of Statistics, University of Toronto, 1993.
15. C. J. Merz, "Combining classifiers using correspondence analysis," in *Advances in Neural Information Processing Systems*, M. I. Jordan, M. J. Kearns, and S. A. Solla, eds., vol. 10, pp. 591–597, The MIT Press, 1998.
16. D. Heckerman and M. P. Wellman, "Bayesian networks," *Communications of the ACM* **38**, pp. 27–30, Mar. 1995.
17. G. Shafer and J. Pearl, eds., *Readings in Uncertain Reasoning*, Morgan Kaufmann, San Mateo, California, 1990.
18. J. Baxter, L. Pratt, and S. Thrun, "A bayesian/information theoretic model of learning to learn via multiple task sampling," *Machine Learning* **28**, pp. 7–40, 1997.
19. P. Domingos and M. Pazzani, "On the optimality of the simple bayesian classifier under zero-one loss," *Machine Learning* **29**, pp. 103–130, 1997.
20. R. N. Dave and K. Bhaswan, "Adaptive fuzzy c-shells clustering and detection of ellipses," *IEEE Transactions on Neural Networks* **3**, pp. 643–662, September 1992.
21. S. Mitra and S. K. Pal, "Fuzzy multi-layer perceptron, inferencing and rule generation," *IEEE Transactions on Neural Networks* **6**, pp. 51–63, January 1995.
22. J.-S. R. Jang, "Self-learning fuzzy controllers based on temporal back propagation," *IEEE Transactions on Neural Networks* **3**, pp. 714–723, September 1992.
23. L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Computation* **4**(6), pp. 888–900, 1992.
24. W. S. Cleveland, E. Grosse, and W. M. Shyu, "Local regression models," in *Statistical Models in S*, J. M. Chambers and T. J. Hastie, eds., pp. 309–376 (of xv + 608), Wadsworth and Brooks/Cole, (Belmont, CA, USA and Pacific Grove, CA, USA), 1991.
25. S. A. Dudani, "The distance-weighted k-nearest-neighbor rule," *IEEE Transactions on Systems, Man and Cybernetics* **6**, pp. 325–327, 1976.
26. L. Breiman, J. H. Friedman, R. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth and Brooks, Pacific Grove, California, 1984.
27. D. Miller and K. Rose, "Hierarchical, unsupervised learning with growing via phase transitions," *Neural Computation* **8**(2), pp. 425–450, 1996.
28. R. A. Jacobs, F. Peng, and M. A. Tanner, "A bayesian approach to model selection in hierarchical mixtures-of-experts architectures," *Neural Networks* **10**(2), pp. 231–241, 1997.
29. M. Kearns, Y. Mansour, A. Y. Ng, and D. Ron, "An experimental and theoretical comparison of model selection methods," *Machine Learning* **27**, pp. 7–50, 1997.
30. S. V. Chakravarthy and J. Ghosh, "Scale-based clustering using the radial basis function network," *IEEE Transactions on Neural Networks* **7**, pp. 1250–1261, September 1996.
31. K. Rose, E. Gurewitz, and G. C. Fox, "A deterministic annealing approach to clustering," Tech. Rep. C3P-857, California Institute of Technology, 1990.
32. R. Murray-Smith and T. A. Johansen, "Local learning in local model networks," in *Multiple Model Approaches to Modelling and Control*, R. Murray-Smith and T. A. Johansen, eds., pp. 185–210, Taylor and Francis, UK, 1997.
33. K. Rose, E. Gurewitz, and G. C. Fox, "Statistical mechanics and phase transitions in clustering," *Physical Review Letters* **65**(8), 1990.
34. R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation* **3**(1), pp. 79–87, 1991.