A Modular Connectionist Architecture For Learning Piecewise Control Strategies

Robert A. Jacobs and Michael I. Jordan*

Department of Brain & Cognitive Sciences
Massachusetts Institute of Technology, Cambridge, MA 02139

Abstract

Methodologies for designing piecewise control laws, such as gain scheduling, are useful because they circumvent the problem of determining a fixed global model of the plant dynamics. Instead, the dynamics are approximated using local models that vary with the plant's operating point. We describe a multi-network, or modular, connectionist architecture that learns to perform control tasks using a piecewise control strategy. The architecture's networks compete to learn the training patterns. As a result, a plant's parameter space is partitioned into a number of regions, and a different network learns a control law in each region.

1. Introduction

When attempting to satisfy the requirements of nonlinear control tasks, it is often impossible or impractical to design continuous control laws that are useful in all the relevant regions of a plant's parameter space. If it is known how the dynamics of a plant change with its operating conditions, then it may be possible to design a piecewise controller that uses different control laws when the plant is operating under different conditions. This principle underlies the design methodology called *gain scheduling*. In a classical gain scheduled design procedure, the designer judiciously selects several operating points which cover the range of the plant's dynamics. At each point, the designer constructs a linear time-invariant approximation to the plant and a linear compensator for the linearized plant. Between operating points the gains of the compensators are interpolated or scheduled. Gain scheduling results in a feedback control system in which the feedback gains are adjusted using feedforward compensation.

An advantage of gain scheduling over more conventional design methodologies is that it circumvents the problem of determining a fixed global model of the plant dynamics. Instead, the designer approximates the plant dynamics using local models that vary with the plant's operating point in a predetermined manner. Unfortunately, the open-loop nature of gain scheduling means that it shares many of the limitations of other model-based feedforward compensators. For many nonlinear plants, accurate models, whether global or local, are hard to formulate and model parameters are difficult to determine. Additionally, such models may be inflexible in the sense that they are closely tied to the particular parameterization of the plant and require significant modification if the plant is altered or if the class of reference trajectories is changed. Consequently, it is often desirable to utilize learning techniques such that feedforward compensators can be implemented without detailed prior knowledge of the plant dynamics.

Nonlinear plants may be difficult to control regardless of whether the control law is designed by a human or learned by a machine. When the control law is designed, gain scheduling is an effective design methodology because it uses task decomposition; the designer using this methodology partitions a complex control task into a number of simpler control tasks. The main claim of this paper is

*This research was supported by a postdoctoral fellowship provided to the first author from the McDonnell-Pew Program in Cognitive Neuroscience, by funding provided to the second author from the Siemens Corporation, and by NSF grant IRI-9013991 awarded to both authors.

that task decomposition is also a useful approach when the control law is learned. We believe that an ideal controller is one that uses local models of the plant dynamics, like gain scheduling controllers, and learns useful control laws despite initial uncertainties about the plant or environment, like learning controllers.

The paper presents a multi-network, or modular, connectionist architecture that learns to perform nonlinear control tasks using a piecewise control strategy. In particular, it uses different networks to learn a plant's inverse dynamics in different regions of its parameter space.

2. A Modular Connectionist Architecture

The technical issues addressed by the modular architecture are twofold: (a) detecting that different training patterns (which are generated when the plant is operating in different regions of its parameter space) belong to different tasks, and (b) allocating different networks to learn the different tasks. These issues are addressed in the architecture by combining aspects of competitive learning and associative learning. Specifically, task decompositions are encouraged by enforcing a competition among the networks comprising the architecture. As a result of the competition, different networks learn different training patterns and, thus, learn to compute different functions. The architecture was first presented in Jacobs, Jordan, Nowlan, and Hinton [6], and combines earlier work on learning task decompositions in a modular architecture by Jacobs, Jordan, and Barto [5] with the mixture models view of competitive learning advocated by Nowlan [9] and Hinton and Nowlan [3].

The architecture, which is illustrated in Figure 1, consists of two types of networks: ***expert networks*** and a ***gating network***. The expert networks compete to learn the training patterns and the gating network mediates this competition. Whereas the expert networks have an arbitrary connectivity, the gating network is restricted to have as many output units as there are expert networks, and the activations of these output units must be nonnegative and sum to one. To meet these constraints, we use the "softmax" activation function (Bridle [2]); specifically, the activation of the i^{th} output unit of the gating network, denoted g_i, is

$$g_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}} \tag{1}$$

where s_i denotes the weighted sum of unit i's inputs and n denotes the number of expert networks. The output of the entire architecture, denoted $\mathbf{y}$, is

$$\mathbf{y} = \sum_{i=1}^{n} g_i \mathbf{y}_i \tag{2}$$

where $\mathbf{y}_i$ denotes the output of the i^{th} expert network. During training, the weights of the expert and gating networks are adjusted simultaneously using the backpropagation algorithm (Rumelhart, Hinton, and Williams [10]) so as to maximize the function

$$\ln L = \ln \sum_{i=1}^{n} g_i e^{-\frac{1}{2\sigma_i^2}\|\mathbf{y}^* - \mathbf{y}_i\|^2} \tag{3}$$

where $\mathbf{y}^*$ denotes the target vector and σ_i^2 denotes a scaling parameter associated with the i^{th} expert network.

This architecture is best understood if it is given a probabilistic interpretation as an "associative gaussian mixture model". Under this interpretation, the training patterns are assumed to be generated by a number of different probabilistic rules. At each time step, a rule is selected with probability g_i and a training pattern is generated by the rule. Each rule is characterized by a statistical model of the form $\mathbf{y}^* = f_i(\mathbf{x}) + \epsilon_i$, where $f_i(\mathbf{x})$ is a fixed nonlinear function of the state vector, denoted $\mathbf{x}$, and ϵ_i is a random variable. If it is assumed that ϵ_i is gaussian with covariance matrix $\sigma_i^2 \mathbf{I}$, then the residual vector $\mathbf{y}^* - \mathbf{y}_i$ is also gaussian and the cost function in Equation 3 is the log likelihood of generating a particular target vector $\mathbf{y}^*$.

The goal of the architecture is to model the distribution of training patterns. This is achieved by gradient ascent in the log likelihood function. To compute the gradient consider first the partial derivative of the log likelihood with respect to the weighted sum s_i at the i^{th} output unit of the gating network. Using the chain rule and Equation 1 we find that this derivative is given by:

$$\frac{\partial \ln L}{\partial s_i} = g(i \mid \mathbf{x}, \mathbf{y}^*) - g_i \tag{4}$$

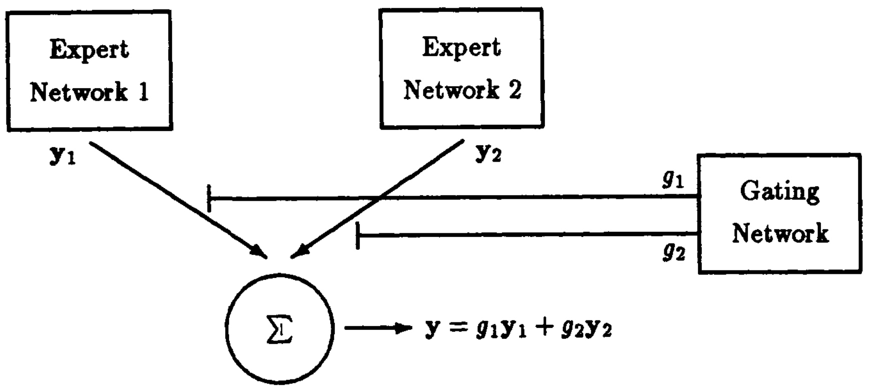

Figure 1: A modular connectionist architecture.

where $g(i \mid \mathbf{x}, \mathbf{y}^*)$ is the a posteriori probability that the i^{th} expert network generates the target vector:

$$g(i \mid \mathbf{x}, \mathbf{y}^*) = \frac{g_i e^{-\frac{1}{2\sigma_i^2}\|\mathbf{y}^* - \mathbf{y}_i\|^2}}{\sum_{j=1}^{n} g_j e^{-\frac{1}{2\sigma_j^2}\|\mathbf{y}^* - \mathbf{y}_j\|^2}}. \qquad (5)$$

Thus the weights of the gating network are adjusted so that the network's outputs—the a priori probabilities g_i—move toward the a posteriori probabilities.

Consider now the gradient of the log likelihood with respect to the output of the i^{th} expert network. Differentiation of $\ln L$ with respect to $\mathbf{y}_i$ yields:

$$\frac{\partial \ln L}{\partial \mathbf{y}_i} = g(i \mid \mathbf{x}, \mathbf{y}^*) \frac{(\mathbf{y}^* - \mathbf{y}_i)}{\sigma_i^2}. \qquad (6)$$

These derivatives involve the error term $\mathbf{y}^* - \mathbf{y}_i$ weighted by the a posteriori probability associated with the i^{th} expert network. Thus the weights of the network are adjusted to correct the error between the output of the i^{th} network and the global target vector, but only in proportion to the a posteriori probability. For each input vector, typically only one expert network has a large a posteriori probability. Consequently, only one expert network tends to learn each training pattern. In general, different expert networks learn different training patterns and, thus, learn to compute different functions.

3. A Multiple Payload Robotics Task

We applied several connectionist systems to the problem of learning a feedforward controller for a simulated two-joint robot arm when a variety of payloads, each of a different mass, must be moved along a specified trajectory.[1] The systems were given the payload's identity (e.g., payload A or payload B) but not its mass. They were trained using a procedure developed by Atkeson and Reinkensmeyer [1], Kawato, Furukawa, and Suzuki [7], and Miller, Glanz, and Kraft [8]. In short, this procedure utilizes a fixed-gain feedback controller to generate training data for approximating a model of the inverse dynamics of the plant. As the inverse dynamic model improves, it generates feedforward terms that improve the trajectory following, thereby providing additional training data in the desired region of state space. This allows the approximation of the inverse dynamic model to be improved further.

Four systems were trained. The systems' inputs were the payload identities and the joint-space positions, velocities, and accelerations of the arm, and their outputs were the feedforward torques that were applied to the arm. The first system, system SN, was a single multi-layer network. The second system, system MA, was a modular architecture with six expert networks and one gating network. The expert networks received the joint variables, and the gating network received the pay-

[1]For a detailed presentation of the application of these systems to the multiple payload robotics task, see Jacobs and Jordan [4].

load identities. The third and fourth systems are referred to as *modular architectures with a share network*. This type of architecture is formed by modifying the standard modular architecture and its use requires additional explanation.

The argument in support of utilizing task decompositions must be tempered by the realization that the members of many sets of tasks are not strictly dissimilar but may contain common features. Consequently, there may be advantages to a system that dedicates one network to learning the common features, and dedicates other networks to learning the features that are unique to each task. The modular architecture illustrated in Figure 2 is such a system.[2] It consists of a *share network* as well as a set of expert networks and a gating network. During training, the share network learns a strategy that is useful for performing all tasks. This strategy is referred to as a shared strategy. The expert networks learn modifications to the shared strategy that are particular to individual tasks. The output of the architecture, $\mathbf{y}$, is the sum of the output of the share network, $\mathbf{y}_s$, and the gated outputs of the expert networks:

$$\mathbf{y} = \mathbf{y}_s + \sum_{i=1}^{n} g_i \mathbf{y}_i. \qquad (7)$$

The training procedure for this architecture is identical to the training procedure for the standard modular architecture with the exception that the networks' weights are adjusted so as to maximize the function

$$\ln L = \ln \sum_{i=1}^{n} g_i e^{-\frac{1}{2\sigma_i^2}\|\mathbf{y}^* - (\mathbf{y}_s + \mathbf{y}_i)\|^2}, \qquad (8)$$

in which the shared term $\mathbf{y}_s$ appears as a component of each of the error terms in the mixture.

When controlling a robot arm to move a variety of payloads, there are many possible decompositions of the task into a shared strategy and a set of modifications to this strategy. The first modular architecture with a share network that we simulated, system MAS, was unconstrained as to the decomposition it could discover. It had a share network, six expert networks, and a gating network. The share and expert networks received the kinematic joint variables, and the gating network received the payload identities. A second modular architecture with a share network—system CMAS—was constrained to discover a particular type of decomposition. Specifically, it was constrained so that the share network learns to produce the correct feedforward torques to control the arm with no payload, and the expert networks learned to add the extra torques required to compensate for the payloads' masses. This constraint was imposed by setting one of the expert network's output to be the zero vector, and fixing the gating network's weights so that this expert network was "gated on" in the absence of a payload.

The learning curves for the four connectionist systems that we simulated are shown in Figure 3. The horizontal axis gives the training time in epochs. The vertical axis gives the joint root mean square error (RMSE) in radians averaged over 25 runs. The curves show that the modular architectures MA, MAS, and CMAS learned significantly faster than the single network SN (at epoch 5, the difference between the performance of any of the modular architectures and the single network is statistically significant at the $p < 0.01$ level). The modular architectures MA, MAS, and CMAS showed similar rates of learning (at epoch 5, none of the pairwise differences between the performances of the modular architectures is statistically significant at the $p < 0.01$ level). The superior rates of learning of the modular architectures over the single network are due to their use of piecewise control strategies. That is, the modular architectures learn to dedicate different expert networks to model the robot arm's inverse dynamics at different operating points.

There are at least three ways that the modular architectures could allocate their expert networks so as to learn to control the robot arm with different payloads. One of the expert networks could learn the appropriate feedforward torques for all payloads. Clearly, this solution doesn't show any task decomposition. A second possibility is that the architectures could use a different expert network to learn the appropriate torques to control the arm with each of the different payloads. This solution shows extensive task decomposition, but fails to take advantage of the similarities between the control laws needed for payloads of similar masses. A third and possibly best solution would be one in which different expert networks learn the appropriate control laws for payloads from different mass categories. For example, one expert network may

[2] A similar system was proposed, but never implemented, by Kawato, Furukawa, and Suzuki [7].

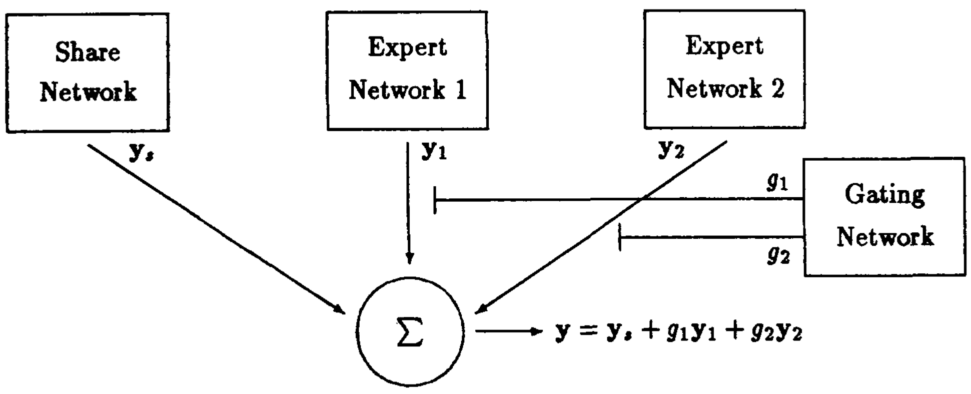

Figure 2: A modular architecture with a share network.

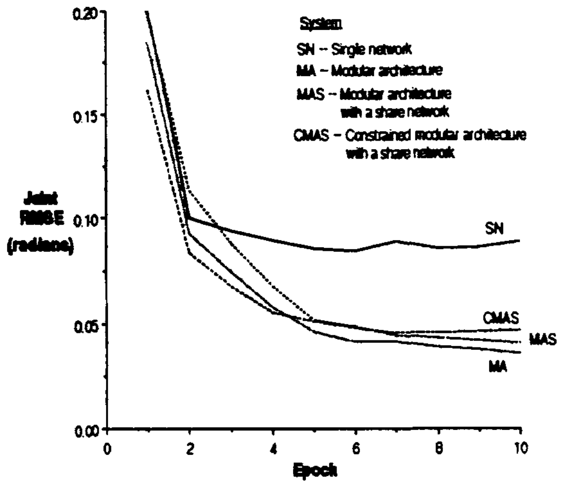

Figure 3: Learning curves for the systems trained in the multiple payload robotics experiment.

learn to control the arm with light payloads, a second expert network may learn to control the arm with medium weight payloads, and a third expert network may learn to control the arm with heavy payloads. This solution takes advantage of task decomposition and of the similarities between the control laws needed for payloads of similar masses. Simulation results show that the third possible solution is the one that is always achieved.

After training the systems as described above, we conducted an additional experiment in which the systems were tested for their ability to learn to control the robot arm with a novel payload. The training period was a single traversal of the desired trajectory with the novel payload. Figure 4 shows the joint root mean square error in radians for the four systems. Based on their experiences with the previous payloads, all the systems performed well. In general, the modular architectures performed better than the single network (the differences in performances between the single network SN and the modular architectures MA and MAS are statistically significant at the $p < 0.01$ level; the performance of CMAS is not significantly different from that of the other systems). Because the expert networks learned local models of the robot arm's inverse dynamics at different operating points, it is surprising that the modular architectures achieve such good generalization to the novel payload. However, an examination of the gating networks' outputs shows that these architectures utilize a weighted combination of the expert networks' outputs that yields an appropriate control signal for the novel payload. This suggests that a modular architecture can use the functions computed by its expert networks as a set of "basis functions" that the gating network can quickly learn to combine in order to control the arm with a variety of payloads.

In summary, the results show that suitably designed modular architectures can learn to perform nonlinear control tasks using a piecewise control strategy. The architecture's networks compete to learn the training patterns. As a result, a plant's parameter space is partitioned into a number of regions, and a different network learns the plant's inverse dynamics in each region. The use of a piecewise control strategy, at least in the experiments reported here, leads to faster rates of learning than the use of a single global control strategy. Furthermore, the results show that the modular architecture can be easily modified so that one network

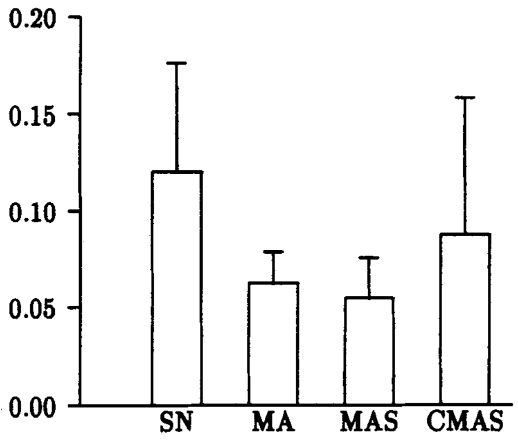

Figure 4: The vertical axis gives the joint root mean square errors in radians on the first trajectory traversal with a novel payload.

learns a shared strategy that is useful at all operating points, and other networks learn modifications to this strategy that are applied in a context sensitive manner.

References

[1] Atkeson, C.G. & Reinkensmeyer, D.J. (1988) Using associative content-addressable memories to control robots. *Proceedings of the IEEE Conference on Decision and Control*, 792–797.

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

[3] Hinton, G.E. & Nowlan, S.J. (1990) The bootstrap Widrow-Hoff rule as a cluster-formation algorithm. *Neural Computation*, 2, 355–362.

[4] Jacobs, R.A. & Jordan, M.I. (1991) Learning piecewise control strategies in a modular connectionist architecture. Submitted to *IEEE Transactions on Neural Networks.*

[5] Jacobs, R.A., Jordan, M.I., & Barto, A.G. (1991) Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks. *Cognitive Science*, in press.

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

[7] Kawato, M., Furukawa, K., & Suzuki, R. (1987) Hierarchical neural-network model for control and learning of voluntary movement. *Biological Cybernetics*, 57, 169–185.

[8] Miller, W.T., Glanz, F.H., & Kraft, L.G. (1987) Application of a general learning algorithm to the control of robotic manipulators. *The International Journal of Robotics Research*, 6, 84–98.

[9] Nowlan, S.J. (1990) Maximum likelihood competitive learning. In D.S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2.* San Mateo, CA: Morgan Kaufmann Publishers.

[10] Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986) Learning internal representations by error propagation. In D.E. Rumelhart, J.L. McClelland, & the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* Cambridge, MA: The MIT Press.