

Local Feedforward Networks

Scott Weaver^{1,2}, Leemon Baird³, Marios Polycarpou¹

¹Department of Electrical and Computer Engineering
University of Cincinnati
Cincinnati, Ohio 45221-0030
Email: `scott.weaver@uc.edu`

²Wright-Patterson Air Force Base
WL/AAAT Bldg 635
2185 Avionics Circle
WPAFB, OH 45433-7301

³United States Air Force Academy
HQ USAFA/DFCS
2354 Fairchild Dr. Suite 6K41
USAFA, CO 80840-6234

Abstract

Although feedforward neural networks are well suited to function approximation, in some applications networks experience problems when learning a desired function. One problem is *interference* which occurs when learning in one area of the input space causes unlearning in another area. Networks that are less susceptible to interference are referred to as spatially *local* networks. To understand these properties, a theoretical framework, consisting of a measure of interference and a measure of network localization, is developed that incorporates not only the network weights and architecture but also the learning algorithm. Using this framework to analyze sigmoidal multi-layer perceptron (MLP) networks that employ the back-prop learning algorithm, we address a familiar misconception that sigmoidal networks are inherently non-local by demonstrating that given a sufficiently large number of adjustable parameters, sigmoidal MLPs can be made arbitrarily local while retaining the ability to represent any continuous function on a compact domain.

1 Introduction

In order for the universal-approximation ability of feedforward neural networks to be useful or even meaningful for function approximation, there needs to be an underlying goodness measure describing how well the network is approximating a desired function [1]. Goodness measures, such as mean squared error, exist and are well used and understood. After a network has been trained, its generalization ability can be measured using Vapnik-Chervonenkis (VC) Dimension to predict the network's accuracy of responses to novel stimuli [2]. Neither ability, however, measures what happens during the learning process; neither the approximation ability nor generalization ability depends upon the learning algorithm. Even though much research is aimed at developing efficient learning algorithms, there are few analytical tools to help us understand the learning process.

To develop these tools, we investigate incremental supervised learning where weight updates are performed after each presentation of a training sample. Training, at one point of the input space, that affects other areas of the input space in an undesirable way is called interference. Analytical tools for investigating, understanding, and perhaps mitigating the problem of interference would be useful because interference problems have been uncovered in various forms by researchers in many areas [3, 4]. For example, consider a dynamical system after it settles into a desired trajectory (where only a small portion of the input/output (i/o) map is used). Without noise, a network function approximator learns the trajectory, reduces the approximation error and then ceases learning. With noise, however, the learning algorithm stays active and continually memorizes the trajectory (because the error never goes to zero) even though there is no need to do so. "Global Network Collapse" results, as other areas of the mapping gradually unlearn due to interference [4]. Another variant of the interference problem is in the classification literature; "Catastrophic Interference" occurs when the training of a new pattern causes the unlearning of originally trained patterns [3]. These and other interference problems may appear different when embedded in their particular applications but the root of these problems is the same; learning tends to interfere with previous learning elsewhere in the input space.

These problems would disappear if learning at a point x in the input space only affected the function mapping at x , as is the case with look-up tables. Of course, taken to this extreme the memory requirement would become prohibitive. When using neural networks, however, learning at x typically does affect the output at $x' \neq x$, leading to the potential of interference [5]. Networks that are less prone to interference are called *local* networks because learning in one region, of the input space causes changes in the i/o map in only a region local to the point of training. The mapping in one region of the input space is less likely to be unlearned when learning migrates to another area of the input space.

With the exception of Baker and Farrell [5] who describe a rule-of-thumb for characterizing useful local networks, the majority of the neural-network literature provides only a cursory look at what it means to be local. We offer a rigorous definition of interference between learning at two points. This measure¹ answers the question “How much does learning at x affect the i/o map at $x' \neq x$?” A second measure proposed in this paper quantifies the localization of a network, by averaging interference over an input domain; this measure describes the network’s ability to resist interference. Unlike previous descriptions of local networks, this measure of localization incorporates the learning algorithm, which is appropriate because the learning algorithm controls how learning occurs, and learning is the cause of interference, which, in turn, is used in the definition of localization.

In Section 2 we define interference and localization which is used in Section 3 to evaluate radial basis functions (RBF) and multi-layer perceptron (MLP) networks with sigmoidal activation functions. We show that the degree of localization of a network is dependent on the network weights as well as the network architecture. In fact, with a sufficient number of carefully chosen weights an MLP can be made as local as desired without losing its universal-approximation ability as shown in Section 4.

2 Definition of Interference and Localization

Consider a network whose input/output mapping is described by $y = f(\mathbf{x}, \mathbf{w})$, where y is the output of the network, $\mathbf{x} \in \mathcal{X}$ is the input, $\mathbf{w} \in \mathcal{W}$ is the weight vector, $f : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ is a smooth mapping describing the network, $\mathcal{X} \subset \mathbb{R}^n$ is the input domain, and $\mathcal{W} \subset \mathbb{R}^m$ is the weight domain. During supervised learning, the objective is to adjust \mathbf{w} such that the network approximates a desired function $y^* = f^*(\mathbf{x})$. We assume the learning algorithm has the form: $\Delta \mathbf{w} = \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e)$, where $\alpha > 0$ is the learning rate constant, \mathbf{H} is the direction for weight change, and $e = y - y^*$ is the approximation error. For notational simplicity we consider single-output networks; extending the results in this paper to multiple-output functions is straight-forward.

Typically, local networks are described using the sensitivity of the network output with respect to weight perturbations. In general, this sensitivity function is insufficient for characterizing interference because it ignores the learning algorithm, which is what is responsible for the weight perturbations. To incorporate the learning algorithm into a measure of interference, consider what happens during one weight update. Given a training input/output sample (\mathbf{x}, y^*) , the current weight \mathbf{w} is updated to a new weight $\mathbf{w} + \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e)$. At the point \mathbf{x} , where the network is trained, the network mapping changes from $f(\mathbf{x}, \mathbf{w})$ to $f(\mathbf{x}, \mathbf{w} + \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e))$. During this weight update the network i/o mapping is affected at other points such as $\mathbf{x}' \neq \mathbf{x}$. To formulate a measure of interference

¹The term “measure” found throughout this paper is not used in the mathematical sense.

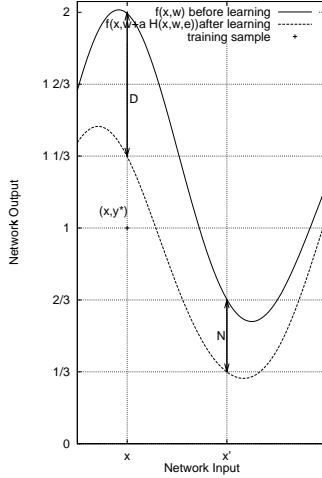


Figure 1: Learning at x causes interference at x' .

at \mathbf{x}' due to learning at \mathbf{x} , we consider the ratio ρ of the change in the i/o map at \mathbf{x}' divided by the change in the i/o map at \mathbf{x} due to learning at \mathbf{x} ; i.e.,

$$\rho \triangleq \frac{f(\mathbf{x}', \mathbf{w}) - f(\mathbf{x}', \mathbf{w} + \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e))}{f(\mathbf{x}, \mathbf{w}) - f(\mathbf{x}, \mathbf{w} + \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e))}. \quad (1)$$

Figure 1 graphically depicts a network's output before and after the weight update at \mathbf{x} . The ratio in (1) is constructed by dividing the change in the output at \mathbf{x}' , which is $N = \frac{1}{3}$, by the change in output at \mathbf{x} , which is $D = \frac{2}{3}$. The ratio $\rho = N/D = 1/2$ is a scalar quantity that represents how much learning at \mathbf{x} interferes with what is known at \mathbf{x}' .

In general, ρ is not a useful measure of interference because (i) ρ depends on an arbitrary learning rate α , (ii) ρ depends on an arbitrary desired training sample (\mathbf{x}, y^*) via e , and (iii) ρ is undefined when the denominator is zero. To redress the first two deficiencies, we take the limit of ρ as α approaches zero and normalize e to one. The choice of e does not affect $\lim_{\alpha \rightarrow 0} \rho$ for algorithms such as back-prop on the standard quadratic cost function, $J = \frac{1}{2}e^2$, resulting in the weight change $\Delta \mathbf{w} = -\alpha e \nabla_{\mathbf{w}} f(\mathbf{x}, \mathbf{w})$. In this case, the particular choice of e is irrelevant because it can be subsumed into α , which approaches zero. Finally, if the $\lim_{\alpha \rightarrow 0} \rho|_{e=1}$ is undefined we define interference to be zero because (the attempt at) learning at \mathbf{x} does not affect the output at \mathbf{x}' . Based on the above modifications, a general definition of interference is given.

Definition 1 Let f represent a network i/o mapping with weight vector \mathbf{w} which is updated according to the learning algorithm $\Delta \mathbf{w} = \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e)$. Then

the interference at \mathbf{x}' due to learning at \mathbf{x} is denoted by $\mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x},\mathbf{x}')$ and is defined as

$$\mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x},\mathbf{x}') \triangleq \begin{cases} \lim_{\alpha \rightarrow 0} \frac{f(\mathbf{x}',\mathbf{w}) - f(\mathbf{x}',\mathbf{w} + \alpha\mathbf{H}(\mathbf{x},\mathbf{w},1))}{f(\mathbf{x},\mathbf{w}) - f(\mathbf{x},\mathbf{w} + \alpha\mathbf{H}(\mathbf{x},\mathbf{w},1))} & \text{if limit exists} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

Interference, according to this definition, is a ratio of the change in the (network) output at \mathbf{x}' divided by the change in the output at \mathbf{x} due to learning at \mathbf{x} . Some miscellaneous notes are: I can be negative, which represents opposite signs for the change in the output at \mathbf{x} and \mathbf{x}' ; when $\mathbf{x} = \mathbf{x}'$ the interference $\mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x},\mathbf{x}') = 1$; and in general, $\mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x},\mathbf{x}') \neq \mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x}',\mathbf{x})$ indicating a non-symmetric behavior between learning at \mathbf{x} and learning at \mathbf{x}' .

For a network function approximator, f , with a well defined gradient in \mathcal{X} , applying L'Hospital's rule to (2) leads to an equivalent yet simpler form:

$$\begin{aligned} \lim_{\alpha \rightarrow 0} \frac{f(\mathbf{x}',\mathbf{w}) - f(\mathbf{x}',\mathbf{w} + \alpha\mathbf{H}(\mathbf{x},\mathbf{w},1))}{f(\mathbf{x},\mathbf{w}) - f(\mathbf{x},\mathbf{w} + \alpha\mathbf{H}(\mathbf{x},\mathbf{w},1))} &= \\ \lim_{\alpha \rightarrow 0} \frac{\frac{\partial}{\partial \alpha}[f(\mathbf{x}',\mathbf{w}) - f(\mathbf{x}',\mathbf{w} + \alpha\mathbf{H}(\mathbf{x},\mathbf{w},1))]}{\frac{\partial}{\partial \alpha}[f(\mathbf{x},\mathbf{w}) - f(\mathbf{x},\mathbf{w} + \alpha\mathbf{H}(\mathbf{x},\mathbf{w},1))]} &= \\ \frac{\left(\nabla_{\mathbf{w}} f(\mathbf{x}',\mathbf{w}) \cdot \mathbf{H}(\mathbf{x},\mathbf{w},1) \right)}{\left(\nabla_{\mathbf{w}} f(\mathbf{x},\mathbf{w}) \cdot \mathbf{H}(\mathbf{x},\mathbf{w},1) \right)} & \end{aligned}$$

where $\nabla_{\mathbf{w}} f(\mathbf{x},\mathbf{w})$ is the gradient vector of $f(\mathbf{x},\mathbf{w})$ with respect to \mathbf{w} . In the special case that the learning algorithm is back-prop (applied to the standard quadratic cost function), $\mathbf{H}(\mathbf{x},\mathbf{w},e) = -e\nabla_{\mathbf{w}} f(\mathbf{x},\mathbf{w})$ and (2) reduces to

$$\mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x},\mathbf{x}') = \begin{cases} \frac{\nabla_{\mathbf{w}} f(\mathbf{x},\mathbf{w}) \cdot \nabla_{\mathbf{w}} f(\mathbf{x}',\mathbf{w})}{\|\nabla_{\mathbf{w}} f(\mathbf{x},\mathbf{w})\|^2} & \text{if } \|\nabla_{\mathbf{w}} f(\mathbf{x},\mathbf{w})\|^2 \neq 0 \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

With this general definition of interference, we now define network localization which provides a measure of how immune a network is to interference.

Definition 2 Let f represent a network i/o mapping with weight vector \mathbf{w} which is updated according to the learning algorithm $\Delta\mathbf{w} = \alpha\mathbf{H}(\mathbf{x},\mathbf{w},e)$. Then the localization of the network over an input domain \mathcal{X} is denoted by $L_{f,\mathbf{w},\mathbf{H},\mathcal{X}}$ and is defined as

$$L_{f,\mathbf{w},\mathbf{H},\mathcal{X}} \triangleq E[\mathcal{I}_{f,\mathbf{w},\mathbf{H}}(\mathbf{x},\mathbf{x}')^2] \quad (4)$$

where $E[\cdot]$ is the expected value over all \mathbf{x} and \mathbf{x}' chosen from some probability density function (pdf) over the input domain \mathcal{X} .

If the pdf of \mathbf{x} and \mathbf{x}' are uniformly distributed, the expected value becomes a double integral with respect to \mathbf{x} and \mathbf{x}' . Small values of $L_{f, \mathbf{w}, \mathbf{H}, \mathcal{X}}$ indicate the network is more immune to interference over the domain \mathcal{X} . This definition transforms a measure of interference (between two points in the input domain) into a measure of localization (of a network over the entire input domain). It is worth emphasizing that these definitions incorporate the learning algorithm. This ensures that different learning algorithms produce different measures of localization, which is appropriate because different algorithms cause different amounts of interference.

3 Application of the Theory of Localization

The previous section offered definitions that were based on the learning algorithm. But it should also be clear that these definitions are also a function of the network architecture and weights. We demonstrate this with the following simple example which confirms our intuition that decreasing the widths (dilation) of a Gaussian RBF increases its degree of localization.

Example 1: Consider a single-input, single-output RBF network with 8 nodes, each of which has an adjustable amplitude a_i , width b_i , and center c_i . The learning algorithm is back-prop (gradient descent on the standard quadratic cost function) and the network has the form

$$f(x, \mathbf{w}) = \sum_{i=1}^8 a_i e^{-\left(\frac{x-c_i}{b_i}\right)^2} \quad (5)$$

where $\mathbf{w} = [a_1 \cdots a_8 \ b_1 \cdots b_8 \ c_1 \cdots c_8]^T$, and $a_i = 1$, $b_i = 0.8$, $c_i = i/8$, for $i \in \{1, \dots, 8\}$. The centers are equally spaced along the input domain $\mathcal{X} = [0, 1]$. Assuming the input x is chosen from a uniform distribution over \mathcal{X} , the measure of localization for this network can be numerically computed as $L_{f, \mathbf{w}, \mathbf{H}, \mathcal{X}} = 0.67$. The localization measure, using a new weight vector

$$\bar{\mathbf{w}} = [a_1 \cdots a_8 \ \frac{b_1}{2} \cdots \frac{b_8}{2} \ c_1 \cdots c_8]^T \quad (6)$$

whose width weights are one-half the size of those in \mathbf{w} , is $L_{f, \bar{\mathbf{w}}, \mathbf{H}, \mathcal{X}} = 0.40$. According to this paper's definition of localization, $f(\mathbf{x}, \bar{\mathbf{w}})$ is more local. \diamond

Although these results are not surprising, they do lead to interesting questions: Because the weight configuration affects how local an RBF is, would the same be true of an MLP with a sigmoidal activation function? If it is, might an MLP be more local than an RBF (with appropriate weight configurations)? To answer this question, we provide a similar example for MLPs with sigmoidal activation functions.

Example 2: Consider a single-input, single-output sigmoidal MLP network with 8 nodes, each of which has a adjustable weights a_i , b_i , and c_i . The

learning algorithm is back-prop (gradient descent on the standard quadratic cost function) and the network has the form

$$f(x, \mathbf{w}) = \sum_{i=1}^8 a_i (1 + e^{-c_i - b_i x})^{-1} \quad (7)$$

where $\mathbf{w} = [a_1 \cdots a_8 \ b_1 \cdots b_8 \ c_1 \cdots c_8]^T$, and $a_i = 1$, $b_i = 8.0$, $c_i = -i$, for $i \in \{1, \dots, 8\}$. Assuming the input x is chosen from a uniform distribution over \mathcal{X} , the measure of localization for this network can be numerically computed as $L_{f, \mathbf{w}, \mathbf{H}, x} = 0.25$. \diamond

The MLP network in Example 2 is more local than either RBF network in Example 1.

To illustrate what causes a network to be local or non-local, we compare interference in a single-input, single-output MLP and RBF, again assuming the learning algorithm is back-prop. First consider an RBF network where the partials of the output with respect to a representative amplitude weight, width weight, and center weight are shown in Figure 2. According to equations (3) and (4) and by continuity of f , if $|x - x'|$ is small then $\mathcal{I}_{f, \mathbf{w}, \mathbf{H}}(x, x') \approx 1$. If $|x - x'|$ is large then it is not possible for the magnitudes of both $\nabla_{\mathbf{w}} f(x, \mathbf{w})$ and $\nabla_{\mathbf{w}} f(x', \mathbf{w})$ to be large (see Figure 2), therefore $\nabla_{\mathbf{w}} f(x, \mathbf{w}) \cdot \nabla_{\mathbf{w}} f(x', \mathbf{w})$ will be small. The term $\nabla_{\mathbf{w}} f(x, \mathbf{w})$ will not be near zero, however, if for every x there is a corresponding element in $\nabla_{\mathbf{w}} f(x, \mathbf{w})$ whose magnitude is large, a reasonable assumption for a useful network² For bases whose widths are small we see according to (3) interference is small and that a small perturbation in a single weight affects only a small portion of the i/o map. As width weights are decreased one can see how an RBF network becomes more local as shown in Example 1.

Repeating this analysis for MLPs leads to Figure 3. Figure 3 (a) shows non-local properties because it is possible for $\nabla_{\mathbf{w}} f(x, \mathbf{w}) \cdot \nabla_{\mathbf{w}} f(x', \mathbf{w})$ to be large even when x and x' are far from one another. The functions shown in Figure 3 (b) and 3 (c), however, vanish rapidly at infinity and indicate local properties (Sjoberg et al. also characterize local networks in this way [6]). From this analysis we see that the network architecture and weights help determine the interference that occurs and hence plays an important role in determining network localization.

4 Localization and Approximation

In the preceding section only the local properties of a network was discussed. Because these networks are operating as function approximators it is imperative that the issue of approximation be addressed in conjunction with localization.

²Baker and Farrell use the term “coverage” for this condition [5].

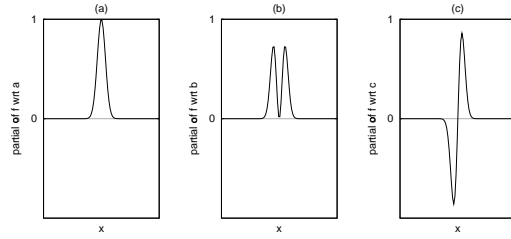


Figure 2: Radial Basis Function partials of the output $f = \sum_{i=1}^N a_i e^{-\frac{(x-c_i)^2}{b_i^2}}$ with respect to an amplitude, width, and center.

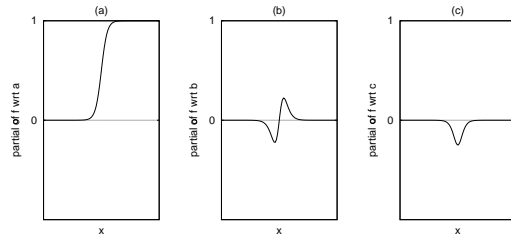


Figure 3: Sigmoidal network partials of the output $f = \sum_{i=1}^N a_i (1 + e^{-c_i - b_i x})^{-1}$ with respect to an amplitude, steepness (at the center), and center.

Theorem 1 helps answer the following question: “Can we find an MLP network that is arbitrarily local *and* approximates any smooth function arbitrarily close in a compact set?”

Theorem 1 *Let \mathcal{X} be a compact subset of \mathbb{R}^n , $\mathbf{H} = -e\nabla_{\mathbf{w}}f(\mathbf{x}, \mathbf{w})$, and $g^*(\mathbf{x})$ be a real valued continuous function on \mathcal{X} . Then for arbitrary $\epsilon_1 > 0, \epsilon_2 > 0$, there exist an integer N and real constants $a_i, c_i (i = 1, \dots, N)$, and $b_{ij} (i = 1, \dots, N) (j = 1, \dots, n)$, such that $h(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^N a_i \left(1 + e^{-c_i - \sum_{j=1}^n b_{ij}x_j}\right)^{-1}$ satisfies*

$$\max_{\mathbf{x} \in \mathcal{X}} |h(\mathbf{x}, \mathbf{w}) - g^*(\mathbf{x})| < \epsilon_1 \quad (8)$$

and

$$L_{h, \mathbf{w}, \mathbf{H}, \mathcal{X}} < \epsilon_2. \quad (9)$$

The proof of the theorem has been omitted due to space limitations, but can be found in Weaver et al. [7]. This theorem corrects the generally held misconception that MLPs are inherently non-local.

5 Conclusion

The formal definitions of interference and localization provided in this paper comprise an analytical framework for investigating local properties of neural networks. This framework is consistent with the neural-network literature, which loosely describes local networks as having an association between a region of the input space and a weight. This characteristic is easy to see in certain networks such as CMAC [8] in which the input space is partitioned and each partition is tied to n weights. When n is 1, and each possible input value has its own partition, the CMAC network collapses to a lookup table, which is the most local network because the learning of new data does not have side affects, i.e., interference doesn’t occur.

The other extreme is not as easy to visualize because interference, according to our definition, can be infinite, and it is hard to imagine a network with infinite interference every where on the input domain. A characteristic of such a non-local network, however, is that learning at each point in the input domain affects the entire i/o map. Neural networks are versatile because they can be either local or non-local [9].

To measure interference, which is a result of learning, one requires the “state” of the network in the form of the architecture and the weights, and one requires the “action” that is occurring, i.e., the learning algorithm. Our measure is applicable whenever we have a continuous network of the form $f(\mathbf{x}, \mathbf{w})$ and a learning algorithm of the form $\Delta \mathbf{w} = \alpha \mathbf{H}(\mathbf{x}, \mathbf{w}, e)$.

Examples 1 and 2 show the importance of the weights in determining the localization of a network. Theorem 1 shows that, assuming back-prop is the learning algorithm, a sigmoidal network can be made to approximate any desired continuous function and be arbitrarily local, provided an arbitrary number of weights are available.

Because different weights produce different amounts of interference, a new learning algorithm might use a network's extra degrees of freedom (in the form of extra weights) to make the network more local, while simultaneously approximating a desired function. In certain applications where interference may cause problems, this learning algorithm may decrease training speed.

Acknowledgements

This work was partially supported under Task 2312 R1 by the United States Air Force Office of Scientific Research.

References

- [1] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Networks*, vol. 2, pp. 183–192, 1989.
- [2] E. B. Baum and D. Haussler, "What size net gives valid generalization?," *Neural Computation*, vol. 1, pp. 81–90, 1989.
- [3] R. French, "Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference," in *Proceedings of the 16th Annual Cognitive Science Society Conference*, vol. 5, pp. 207–220, 1994.
- [4] D. Sofge and D. White, "Applied learning: optimal control for manufacturing," in *Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches* (D. White and D. Sofge, eds.), (New York, NY), pp. 259–281, Van Nostrand Reinhold, 1992.
- [5] W. Baker and J. Farrell, "An introduction to connectionist learning control systems," in *Handbook of Intelligent Control Neural, Fuzzy, and Adaptive Approaches* (D. White and D. Sofge, eds.), (New York, NY), pp. 35–63, Van Nostrand Reinhold, 1992.
- [6] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: A unified overview," 1995.

- [7] S. Weaver, L. Baird, and M. Polycarpou, “An analytical framework for local feedforward networks,” Tech. Rep. TR 195/07/96/ECECS, University of Cincinnati, 1996.
- [8] J. Albus, “A new approach to manipulator control: The cerebellar model articulation controller,” *Journal of Dynamic Systems, Measurement, and Control*, pp. 220–227, 1975.
- [9] A. G. Barto, “Connectionist learning for control,” in *Neural Networks for Control* (I. W. Thomas Miller, R. S. Sutton, and P. J. Werbos, eds.), (Massachusetts Institute of Technology, MA), pp. 5–58, MIT Press, 1990.