On the Localization of Feedforward Networks

Scott Weaver* **Leemon Baird†** **Marios Polycarpou‡**

*Wright-Patterson Air Force Base
WL/AAAT Bldg 635, 2185 Avionics Circle
WPAFB, OH 45433-7301
scott.weaver@uc.edu

Abstract– Interference in neural networks occurs when learning in one area of the input space causes unlearning in another area. Networks that are less susceptible to interference are called spatially local networks. These networks are often used in neurocontrol, in online applications, where, because of the real time nature of the task, interference is often a problem. Although there are heuristics as to what makes a network local, there is no theoretical framework for measuring localization. This paper provides a formal definition of interference and localization that will allow us to measure a network's local properties. These definitions will be useful in developing learning algorithms that make networks more local. This may lead to faster learning over the entire input domain.

1. Introduction

For some applications, a neural network's ability to generalize may actually hinder its ability to learn. In neurocontrol, for example, the network is usually trained online using an incremental learning algorithm, with desired input/output data as a teaching signal. The data is obtained in real-time from a continuous dynamical system, causing consecutive training samples to be near one another in input space, and hence highly correlated. This situation causes training to be concentrated in one area of the input space allowing other areas to suffer potentially from unlearning, leading to slower learning of the entire function.

Actually, concentration of learning in one area would not be a problem if learning at a point x in the input space only affected the network mapping at x, as is the case with lookup tables. When using neural networks, however, learning at x typically does affect the output at $x' \neq x$. This allows networks to generalize better but it is also responsible for *interference* [1]. When learning at x causes correct learning at x' we say the network is generalizing well, but when learning at x causes incorrect learning a x' we say there is interference between training at the two points. The right balance between generalization and interference is surely different for different applications. It is safe to say, however, that as consecutive training samples become closer to one another, interference becomes more of a problem, a problem that can be serious during neurocontrol. Since we have little control over how the training data is presented (because of the application) and we would like some degree of generalization (because there is a continuum of inputs), making networks have less interference is the next logical step.

Networks that are less likely to have problems with interference are said to be local. One approach for obtaining local networks is to choose a network architecture that has good localization properties, regardless of the weight configuration. Finding the balance between generalization and interference may be a difficult task for the network designer. An alternative method is to choose a learning algorithm that adjusts weights in such a way that local properties emerge from the network during learning. To this end, we need a better understanding of what makes local networks local.

Although the concept of local networks is found in the literature [2], what is lacking is a rigorous definition of network localization. Generalization, a related concept, is well defined in [3] for example, but the theoretical framework assumes a finite number of input-output pairs. We propose a measure of network localization based on the extent of interference during learning. This definition will provide a means for comparing the localization of various networks and possibly pave the way for faster learning algorithms.

2. Localization

Consider a network whose input/output characteristics are described by $y = f(\mathbf{x}, \mathbf{w})$, where $\mathbf{x}$ is the input to the network, $\mathbf{w}$ is the weight vector, y is the output of the network, $f : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}$ is a smooth mapping describing the function implemented by the network, $\mathcal{X} \subset \mathbb{R}^m$ is the input domain, and $\mathcal{W} \subset \mathbb{R}^n$ is the weight domain. During supervised learning, the objective is to adjust $\mathbf{w}$ such that the network approximates a desired function of $\mathbf{x}$, the samples are given by $(\mathbf{x}, y^*)$, where y^* is the exemplar (desired) output.

Most descriptions of localization found in the literature ignore the role of the learning algorithm. However, since the goal of making local networks is to reduce interference and since interference is directly affected by the learning mechanism, we propose a measure of localization based on interference and a measure of interference based on learning.

To incorporate the learning mechanism into the definition of interference consider what happens when the network learns for one time step. The training exemplar $(\mathbf{x}, y^*)$ is chosen to yield an error $y - y^*$ that is normalized to be one. This leads to a choice of $y^* = f(\mathbf{x}, \mathbf{w}) - 1$. After one step the new weight will be $\mathbf{w} + \alpha \mathbf{H}(\mathbf{x}, \mathbf{w})$ where $\alpha \in \mathbb{R}$ is the learning rate, and $\mathbf{H} : \mathcal{X} \times \mathcal{W} \rightarrow \mathbb{R}^n$ specifies

*Partially supported under Task 2312 R1 by the United States Air Force Office of Scientific Research.

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

‡Department of Electrical and Computer Engineering, University of Cincinnati, Cincinnati, Ohio 45221-0030

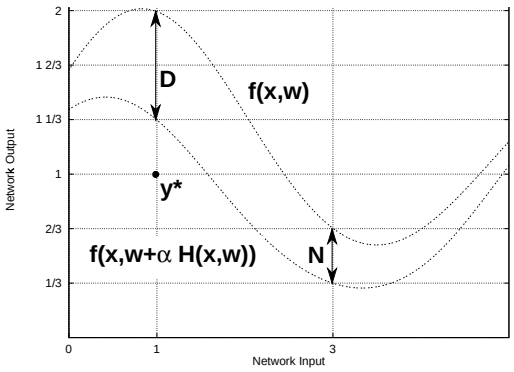

Figure 1: Learning at $\mathbf{x} = 1$ causes interference at $\mathbf{x}' = 3$. The top (bottom) curve is the network mapping before (after) training at $\mathbf{x}$. The desired training exemplar is $(\mathbf{x}, y^*)$.

the direction for weight change.

Definition 1 *The interference at $\mathbf{x}'$ due to learning at $\mathbf{x}$ when using network $f(\mathbf{x}, \mathbf{w})$ and learning algorithm $\mathbf{w} + \alpha\mathbf{H}(\mathbf{x}, \mathbf{w})$ is given by*

$$I_{\mathbf{H},f,\mathbf{W}}(\mathbf{x}, \mathbf{x}') = \lim_{\alpha \to 0} \frac{f(\mathbf{x}', \mathbf{w}) - f(\mathbf{x}', \mathbf{w} + \alpha\mathbf{H}(\mathbf{x}, \mathbf{w}))}{f(\mathbf{x}, \mathbf{w}) - f(\mathbf{x}, \mathbf{w} + \alpha\mathbf{H}(\mathbf{x}, \mathbf{w}))}. \tag{1}$$

The validity of this definition is taken up later in a simpler yet equivalent formulation. Interference, according to the above definition, is a sensitivity of the (network) output at $\mathbf{x}'$ with respect to the output at $\mathbf{x}$ due to learning at $\mathbf{x}$. If training at $\mathbf{x}$ has little affect on the output at $\mathbf{x}'$, the magnitude of the ratio of (1) will be small; if interference occurs, training at $\mathbf{x}$ affects the output significantly at $\mathbf{x}'$ causing the magnitude of the ratio of (1) to be large. Figure 1 graphically depicts the network's output before and after learning at $\mathbf{x}$ for a finite α. To form the ratio in (1), divide the change in output at $\mathbf{x}'$, which is $N = \frac{1}{3}$, by the change in output at $\mathbf{x}$, which is $D = \frac{2}{3}$. If the ratio $\frac{N}{D}$ remains constant, as α approaches zero, $I_{\mathbf{H},f,\mathbf{W}}(\mathbf{x}, \mathbf{x}') = \frac{1}{2}$.

Equation (1) is equivalent to a ratio of two dot products

$$I_{\mathbf{H},f,\mathbf{W}}(\mathbf{x}, \mathbf{x}') = \left(\frac{\nabla_{\mathbf{W}} f(\mathbf{x}', \mathbf{w}) \cdot \mathbf{H}(\mathbf{x}, \mathbf{w})}{\nabla_{\mathbf{W}} f(\mathbf{x}, \mathbf{w}) \cdot \mathbf{H}(\mathbf{x}, \mathbf{w})} \right) \tag{2}$$

where $\nabla_{\mathbf{W}} f(\mathbf{x}, \mathbf{w})$ is the gradient vector of the scalar function $f(\mathbf{x}, \mathbf{w})$ with respect to $\mathbf{w}$. This equation is valid, and hence the above definition of interference is valid, when the quantity $\nabla_{\mathbf{W}} f(\mathbf{x}, \mathbf{w})$ exists and the denominator is non-zero for each $\mathbf{x}$ and $\mathbf{w}$. In the particular case of gradient descent ($\mathbf{H}(\mathbf{x}, \mathbf{w}) = -\nabla_{\mathbf{W}} f(\mathbf{x}, \mathbf{w})$) the later condition is satisfied when the network has a bias weight at the output layer and if the output layer has an activation function, it must be monotonic. Now we are ready to define a measure of network localization.

Definition 2 *The localization of network $f(\mathbf{x}, \mathbf{w})$ using a learning algorithm $\mathbf{w} + \alpha\mathbf{H}(\mathbf{x}, \mathbf{w})$ is given by*

$$L_{\mathbf{H},\mathbf{W}} = E[I_{\mathbf{H},f,\mathbf{W}}(\mathbf{x}, \mathbf{x}')^2]$$

where $E[\cdot]$ is the expected value.

The network's localization $L_{\mathbf{H},\mathbf{W}}$ is a positive quantity. The smaller the value of $L_{\mathbf{H},\mathbf{W}}$, the more local the network.

Example: Let us compute $L_{\mathbf{H},\mathbf{W}}$, the localization of the network $f(\mathbf{x}, \mathbf{w})$ using gradient descent on the quadratic cost function $J(\mathbf{x}, \mathbf{w}) = \frac{1}{2}(f(\mathbf{x}, \mathbf{w}) - y^*)^2$. The network $f(\mathbf{x}, \mathbf{w})$ is a piecewise constant function with domain $\mathcal{X} = [0,\ 1)$. Let $f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{N} w_i d_i(\mathbf{x})$, where for $i \in \{1, 2, \cdots, N\}$ each basis function $d_i(\mathbf{x})$ is defined as

$$d_i(\mathbf{x}) = \begin{cases} 1 & \text{if } \frac{i-1}{N} \leq \mathbf{x} < \frac{i}{N} \\ 0 & \text{otherwise.} \end{cases}$$

If $j \in \{1, 2, \cdot\cdot, N\}$ is such that $\mathbf{x} \in [\frac{j-1}{N}, \frac{j}{N})$ then $\nabla_{\mathbf{W}} f(\mathbf{x}, \mathbf{w}) = [0\ \cdots\ 0\ d_j(\mathbf{x})\ 0\ \cdots\ 0]^T$. If $k \in \{1, 2, \cdot\cdot, N\}$ is such that $\mathbf{x}' \in [\frac{k-1}{N}, \frac{k}{N})$ then $\nabla_{\mathbf{W}} f(\mathbf{x}', \mathbf{w}) = [0\ \cdots\ 0\ d_k(\mathbf{x})\ 0\ \cdots\ 0]^T$ and it follows from (2) and $\mathbf{H}(\mathbf{x}, \mathbf{w}) = -\nabla_{\mathbf{W}} f(\mathbf{x}, \mathbf{w})$ that

$$I_{\mathbf{H},f,\mathbf{W}}(\mathbf{x}, \mathbf{x}') = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise.} \end{cases}$$

When j and k are equal, both $\mathbf{x}$ and $\mathbf{x}'$ fall within the influence of the same basis function leading to interference. If $\mathbf{x}$ and $\mathbf{x}'$ are chosen randomly from $\mathcal{X} = [0,\ 1)$ with a uniform distribution, then $Pr[j = k] = \frac{1}{N}$ and we get $L_{\mathbf{H},\mathbf{W}} = \frac{1}{N}$. One can see that as $N \to \infty$, $L_{\mathbf{H},\mathbf{W}} \to 0$; equivalently, as the number of basis functions increase, for this example, the network becomes more local. $\diamond$

3. Discussion and Conclusion

The proposed measures of localization and interference are applicable to most network/algorithm combinations. They are valuable when comparing networks and evaluating a network's learning capability. The measure of localization allows the development of learning algorithms that find, in real time, a balance between generalization and localization. Preliminary simulations performing gradient descent on a new cost function, one that penalizes poor localization as well as poor approximation, show that with highly correlated input exemplars the new learning mechanism learns faster than with the quadratic error cost function alone. It is important to stress that localization of a network is not a goal in itself. It is merely a tool one can use to accomplish the true goal, which is to reduce the approximation error for all inputs and to do so quickly.

References

[1] 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.

[2] 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.

[3] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*. Redwood City, CA: Addison-Wesley, 1991.