Why Tanh:

Choosing a Sigmoidal Function

Barry L. Kalman

Stan C. Kwasny

Center for Intelligent Computer Systems†
Department of Computer Science
Washington University, Campus Box 1045
St. Louis, Missouri 63130

ABSTRACT

While everyone accepts the need for an activation function in a neural unit, justifications for the choice of a sigmoidal are difficult to find. We argue for the use of the hyperbolic tangent. While the exact shape of the sigmoidal makes little difference once the network is trained, we show that it possesses particular properties that make it appealing for use while training. We further show that by paying attention to scaling, tanh(1.5 x) has the additional advantage of equalizing training over layers. This result can easily generalize to several standard sigmoidal functions commonly in use.

1. Introduction

The activation of a single neural unit in a neural model is determined as a function (activation function) of the net input to that unit. In feed-forward neural networks, a sigmoidal function is generally chosen for that purpose. Other names for this function are the squashing function or the logistic function. We will use the symbol $\sigma(x)$ to indicate such a function.

By definition, a sigmoidal function must be asymptotically bounded from above and below, defined over all real values, and differentiable with a positive derivative everywhere. For example, each of the following is a sigmoidal function:

$$f_1(x) = \frac{1-e^{-x}}{1+e^{-x}} = \tanh(\frac{x}{2}); \quad f_2(x) = \frac{1}{1+e^{-x}} = \frac{f_1(x)+1}{2}; \quad f_3(x) = \frac{2}{\pi}\tan^{-1}(x)$$

As hardware implementations of back propagation and related training algorithms are anticipated, the choice of sigmoidal function should be carefully justified. With several possible candidates along with their variations, it is important to examine which candidates provide the best properties. With some care, choices that would naturally limit the form of the sigmoidal need not eliminate ones with the best properties.

† The sponsors of the Center are McDonnell Douglas Corporation and Southwestern Bell Telephone Company.

0-7803-0559-0 /92 $3.00 © 1992 IEEE

2. How are sigmoidal functions justified?

How does one decide which sigmoidal function to use? In the literature, selection is justified on either neurobiological grounds or mathematical grounds (Williams, 1986) (Jordan, 1986), on pragmatic grounds (Hecht-Nielsen, 1990) (Wasserman, 1989), or, most commonly, on intuitive grounds. Most of the intuition comes from rather vague arguments related to its squashing capabilities and to the fact that the gain (slope) through the axis can be easily manipulated. It is generally believed that "the exact details of the sigmoid are not critical in a back-propagation network." (p.19, Caudill, 1990). While this is true for the performance of an already-trained network, there are important differences that can impact training dramatically. Therefore, attention should focus on choosing an activation function that exhibits the best properties for training. Our arguments come from this perspective.

3. Properties of sigmoidal functions

The purpose of training is to reduce error as calculated by an error function. If p is a pattern and u is an output unit, the form of a generic error function is

$$\Psi = \sum_{p,u} g_{pu}$$

where g_{pu} is a function of a target value for an output unit, t_{pu}, and its activation value, a_{pu}. Often, g_{pu} is chosen as the square of the error, e_{pu}^2, where $e_{pu} = (t_{pu} - a_{pu})$. For effective training, g_{pu} should satisfy the following properties:

(1) If a_{pu} approaches t_{pu} (that is, $e_{pu} \rightarrow 0$), then g_{pu} should approach 0.

(2) If a_{pu} approaches $-t_{pu}$ when $t_{pu} = \pm 1$ or $a_{pu} \rightarrow \pm 1$ when $-1 < t_{pu} < 1$ then g_{pu} should approach ∞.

(3) If a_{pu} approaches t_{pu} then g_{pu}' should approach 0.

(4) If a_{pu} approaches $-t_{pu}$ when $t_{pu} = \pm 1$ or $a_{pu} \rightarrow \pm 1$ when $-1 < t_{pu} < 1$ then $| g_{pu}' |$ should approach ∞.

Properties (1) and (2) apply to g_{pu} while (3) and (4) apply to its derivative, g_{pu}'. For error functions based on the sum of squares, only properties (1) and (3) are satisfied. However, Property (2) is needed to discourage local extrema and saddle points that are exactly opposite to the training value and property (4) is required to affect training algorithms in a way that pushes away from such exactly opposite values. The number of saddle points that are exterior in this sense can be very large (see Kalman & Kwasny, 1991).

4. How to satisfy these properties

Since it is desirable to find error functions that satisfy all four properties, we have investigated a family of scaled error functions of the form $g_{pu} = \frac{e^2_{pu}}{s_{pu}}$. The error computation is therefore:

$$\text{kerr} = \sum_{p,u} \left(\frac{e_{pu}^2}{s_{pu}} \right)$$

Members of such a family of functions represent an easy-to-compute variation on the normal sum of squares functions. The scaling factor can be chosen to make various of the four properties true. If s_{pu} is chosen to be $1-\sigma^2$, then kerr will satisfy properties (1) and (2). If s_{pu} is chosen to be σ', then kerr will satisfy properties (3) and (4). All four conditions are satisfied exactly when

$$\sigma' = 1-\sigma^2$$

This differential equation is only satisfied by

$$\sigma(x) = \tanh(\lambda x)$$

for λ a scalar. Thus, the choice of tanh is seen as advantageous in the construction of efficient training methods. Further details and descriptions can be found in (Kalman & Kwasny, 1991).

5. Scaling learning across layers

An independent argument in favor of tanh concerns the scaling of weight layers in feedforward networks. Rigler et al. (1991) have argued that in networks containing multiple weight layers, the changes to the weights in each layer are scaled disproportionately by the gradient term, δ_{pu}. unless measures are taken to adjust for that. They provide a method which maintains the expected value of $\sigma' = 1$.

For tanh, this expected value for layer n (counting back from the output layer) is given by:

$$E[\, \lambda^n (1 - \sigma^2(x_{pn})) (1 - \sigma^2(x_{p(n-1)}))(1 - \sigma^2(x_{p(n-2)})) \ \ldots \ ]$$

By integrating over the interval [-1,1] and solving for λ , we find that

$$\lambda = \frac{3}{2}$$

maintains the expected value at unity for all weight layers.

Thus, weight layers can be equally scaled by choosing

$$\sigma(x) = \tanh(\frac{3}{2} x)$$

Using this sigmoidal function is particularly valuable in networks with very deep layering. Training is spread uniformly over each layer.

In general, to provide activations within the range [low,high] and to maintain all of the properties and features mentioned above, the sigmoidal should be chosen as

$$\sigma(x) = \frac{(high - low)}{2} \tanh(\frac{3}{2} x) + \frac{(high + low)}{2}$$

Choosing the interval [-1,+1], yields $f_1(3\ x)$. Choosing the interval [0,+1], yields $f_2(3\ x)$.

6. Conclusion

Extensive testing of these methods with various benchmarks supports the conclusion that tanh, with these particular parameters, yields the best overall properties in an activation function for layered feed-forward networks. Thus, we have argued from several independent perspectives that the choice for sigmoidal should be a form of tanh.

References

Caudill, Maureen. (1990). *Neural Networks Primer*. San Francisco: Miller Freeman Publications.

Hecht-Nielsen, Robert. (1990). *Neurocomputing*. Reading, MA: Addison-Wesley.

Jordan, M. I. (1986). An Introduction to Linear Algebra in Parallel Distributed Process. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing*. Cambridge, MA: MIT Press.

Kalman, Barry L., and Stan C. Kwasny. (1991). A Superior Error Function for Training Neural Networks. *Proceedings of the International Joint Conference on Neural Networks*, Vol. 2, Seattle, Washington, 49–52.

Rigler, A.K., J.M. Irvine, and T.P. Vogl. (1991). Rescaling of Variables in Back Propagation Learning. *Neural Networks*, 4, 225-229.

Wasserman, Philip D. (1989). *Neural Computing: Theory and Practice*. New York: Van Nostrand Reinhold.

Williams, R. J. (1986). The Logic of Activation Functions. In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing*. Cambridge, MA: MIT Press.