Noise Robustness of EBNN learning

Ryusuke Masuoka *
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
E-mail: masuoka@cs.cmu.edu

Abstract

A variety of methods have recently been proposed for constraining neural networks to fit various constraints while being trained. One such approach is to constrain the function approximated by the network to fit desired slopes, or derivatives. Such slopes may be provided by the designer, as in Simard's character recognizer network which was constrained so that the slope of the output with respect to translations, rotations, etc. of the input should be zero. Alternatively, target slopes may be generated automatically by program as in Explanation Based Neural Network (EBNN) learning. While slope information is known to improve generalization, sometimes slope information as well as value information is corrupted by noise.

This paper explores the effects of noise in value and slope information on EBNN learning, compared with standard backpropagation. Experimental results show several characteristics of noise robustness of EBNN learning.

1 Introduction

Among learning tasks that fall into the category of function approximation, there are some for which slope information about the target function is available. Previous work in Explanation Based Neural Network (EBNN) learning ([Mitchell and Thrun, to appear]) and in learning the invariance of pattern recognition with respect to a transformation group ([Simard *et al.*, 1992]) has demonstrated the feasibility of exploiting slope information. Detailed mathematical and experimental analyses of these methods are given in [Masuoka *et al.*, 1993].

But for tasks like EBNN learning where slope information is created from less or no prior knowledge, noise in slope information is inevitable. The effects of such noise have to be determined.

The goal of this paper is to determine the effects of noise on EBNN learning empirically. First we give a very quick overview of EBNN learning. Then we devise a learning task of approximating a two-variable nonlinear function and compare the performance of EBNN learning with that of standard backpropagation. We also determine the characteristics of EBNN learning with respect to the noise levels of both value and slope information.

2 Overview of EBNN learning

In this section, we give a very quick overview of EBNN learning along with backpropagation. The details of the algorithms of EBNN learning are given in [Masuoka *et al.*, 1993].

The definitions in Table 1 are used throughout this paper. The squashing function of each unit is $(1 + e^{-z})^{-1}$. We denote by β_i^k (*output Jacobian*), the derivative of the activation of the output unit k with respect to the activation of the input unit i. (i.e. $\beta_i^k = \partial x_k / \partial x_i$.) We also denote by $\tilde{\beta}_i^k$ (*desired output Jacobian*), the desired derivative of the activation of the output unit k with respect to the activation of the input unit i.

We distinguish two types of errors: The *value error* (Equation 1) and the *EBNN slope error* (Equation 2).

$$E_{\mathrm{v}} \quad := \quad \frac{1}{2} \sum_{p\,:\,\text{pattern}} \sum_{k\,:\,\text{output}} (\tilde{x}_k - x_k)^2 \tag{1}$$

*Visiting Scientist From Fujitsu Laboratories Ltd., Kanagawa, Japan

x_i	*activation* of unit i
$\tilde{x}_k$	*desired output value* of output unit k
w_{ij}	*weight* from unit j to unit i
E_{bp}, E_{ebnn}	*total errors* for backpropagation and EBNN learning
E_{v}, $E_{\text{ebnn-s}}$	*value and EBNN slope errors*
$\Delta_{\text{v}} w_{ij}$	*weight change* with respect to value error
$\Delta_{\text{ebnn-s}} w_{ij}$	*weight change* with respect to EBNN slope error
$\overline{\Delta} w_{ij}$	*last (total) weight change*
$\eta_{\text{bp-v}}$	*stepsize* of backpropagation for value error
$\eta_{\text{ebnn-v}}$	*stepsize* of EBNN learning for value error
$\eta_{\text{ebnn-s}}$	*stepsize* of EBNN learning for EBNN slope error
α	*momentum coefficient*

Table 1: **Definitions**

$$E_{\text{ebnn-s}} := \frac{1}{2} \sum_{p\,:\,\text{pattern}} \sum_{k\,:\,\text{output}} \sum_{i\,:\,\text{input}} (\tilde{\beta}_i^k - \beta_i^k)^2 \tag{2}$$

For the sake of clarity we do *not* label any value with its pattern label p.

In EBNN learning, both the EBNN slope error function along with value error function used in standard backpropagation are used to train the neural networks. While the total error for standard backpropagation is given by Equation 3, the total error for EBNN learning is given by Equation 4.

$$E_{\text{bp}} = \eta_{\text{bp-v}} E_{\text{v}} \tag{3}$$

$$E_{\text{ebnn}} = \eta_{\text{ebnn-v}} E_{\text{v}} + \eta_{\text{ebnn-s}} E_{\text{ebnn-s}} \tag{4}$$

We set the weight changes with respect to each energy as follows.

$$\Delta_{\text{v}} w_{lj} = -\frac{\partial E_{\text{v}}}{\partial w_{lj}} \tag{5}$$

$$\Delta_{\text{ebnn-s}} w_{lj} = -\frac{\partial E_{\text{ebnn-s}}}{\partial w_{lj}} \tag{6}$$

Then we update the weights as follows. The updating rule for ordinary back propagation is is given by Equation 7 and that for EBNN learning is given by Equation 8.

$$w_{lj} \longleftarrow w_{lj} + \eta_{\text{bp-v}} \Delta_{\text{v}} w_{lj} + \alpha \overline{\Delta} w_{lj} \tag{7}$$

$$w_{lj} \longleftarrow w_{lj} + \eta_{\text{ebnn-v}} \Delta_{\text{v}} w_{lj} + \eta_{\text{ebnn-s}} \Delta_{\text{ebnn-s}} w_{lj} + \alpha \overline{\Delta} w_{lj} \tag{8}$$

3 Experiments

We use the following two-variable nonlinear function for a learning problem. (See Figure 4.)

$$g(x, y) = 0.5 e^{-10((x-0.2)^2+(y-0.5)^2)} + 0.25 \sin(8xy) + 0.25 \tag{9}$$

The task is to approximate this function. The neural network used in the experiments consists of two input units (for x and y), six hidden units, and one output unit (for $z = g(x, y)$). The network is fully connected between layers and each unit except input units has a bias.

One pattern for standard backpropagation consists of two input values, x and y, and one desired output value, $z = g(x, y)$, for the output unit. One pattern for EBNN learning consists of two input values, x and y, one desired output value, $g(x, y)$, and two desired output Jacobians, $\partial g/\partial x\,(x, y)$ and $\partial g/\partial y\,(x, y)$.

The common conditions for the experiments in this paper are the following.

- We use the following parameters for weights and biases update. For backpropagation, $\eta_{\text{bp-v}} = 2$, $\alpha = 0.9$. For EBNN learning: $\eta_{\text{ebnn-v}} = \frac{16}{9}$, $\eta_{\text{ebnn-s}} = \frac{1}{9}$, $\alpha = 0.9$. [1]

[1] These parameters are chosen empirically. (See [Masuoka *et al.*, 1993] for the details.)

- We draw random numbers from the uniform distribution on $[-0.01, 0.01]$ to set initial values of weights and biases. We use the same initial values of weights and biases for all the experiments.
- Ten sets of twenty patterns are drawn randomly from the uniform distribution on $[0, 1] \times [0, 1]$ and they are fixed. When we say the noise level is σ, the random values are drawn from the normal distribution with variation σ and mean 0. This noise is added to the desired output values or desired output Jacobians or both. When the resulting desired output value is less than 0.0 or greater than 1.0, it is reset to 0.0 or 1.0 accordingly.
- We call the noise level for desired output values the *value noise level*, and the noise level for desired output Jacobians the *slope noise level*.
- Each trial is run for 200,000 epochs. Every 200 epochs, the average of square value errors of the network output on the 26×26 equidistant lattice points on $[0, 1] \times [0, 1]$ is calculated. [2]

$$\frac{1}{26 \times 26} \sum_{i=0}^{25} \sum_{j=0}^{25} \left[g\left(\frac{i}{25}, \frac{j}{25}\right) - n\left(\frac{i}{25}, \frac{j}{25}\right) \right]^2 \tag{10}$$

  We call this average the *generalization error*. The best generalization error during 200,000 epochs for each trial is recorded.
- The *performance* for a particular learning method under certain conditions is measured by the average of the best generalization error of the ten trials run for the ten sets of twenty patterns above.

First we run standard backpropagation, plotting performance as the value noise level is increased. See Figure 1.

We then run EBNN learning, plotting performance as the slope noise level is increased (with zero value noise). See Figure 2.

We also run EBNN learning, changing both the value and slope noise levels. See Figure 3.

Empirical observations from this and other experiments are the following:

- EBNN learning generalizes better than standard backpropagation even if value noise is present. (Compare Figure 1 and the cross section of Figure 3 for which *slope noise level* = 0.0.)
- The greater the noise level for desired output values, the less the slope noise level matters. Especially, the level of slope noise for which EBNN learning has the same performance as standard backpropagation increases, as the value noise level increases.

4 Summary

In this paper, we have determined the effects of noise on EBNN learning empirically using a two-variable nonlinear function.

Future work will include experiments using more realistic data.

References

[Masuoka *et al.*, 1993] R. Masuoka, S. Thrun, and T. M. Mitchell. Learning slopes by neural networks. Technical Report to appear, CMU Computer Science, 1993.

[Mitchell and Thrun, to appear] Thomas M. Mitchell and Sebastian Thrun. Explanation based neural network learning for robot control. In Stephen J. Hanson, Jack Cowan, and Lee Giles, editors, *Advances in Neural Information Processing Systems 5*. Morgan Kaufmann, San Mateo, CA, to appear.

[Simard *et al.*, 1992] Patrice Simard, Bernard Victorri, Yann LeCun, and John Denker. Tangent prop – a formalism for specifying selected invariances in an adaptive network. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 895–903. Morgan Kaufmann, San Mateo, CA, 1992.

[2] Here we refer to the function calculated by the neural network as n(x,y).

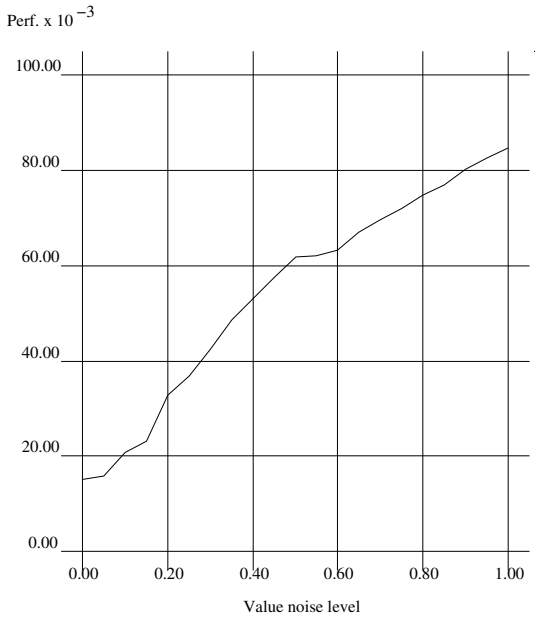

Figure 1: **Performance of backpropagation:** The performance as defined above of standard backpropagation against the value noise level is plotted. Noise is added only to desired output values. The value noise level is varied from 0.0 to 1.0 in increments of 0.05.

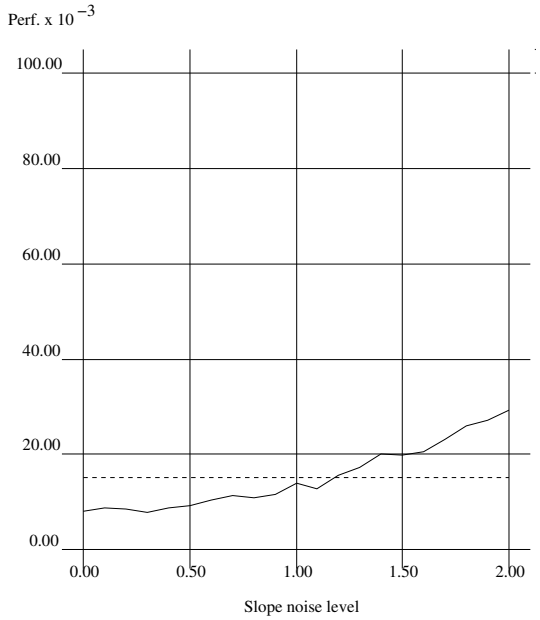

Figure 2: **Performance of EBNN learning (1):** The performance as defined above of EBNN learning against the slope noise level is plotted as the solid line. Noise is added only to desired output Jacobians. The slope noise level is changed from 0.0 to 2.0 with stepsize 0.1. The dotted line is the performance of standard backpropagation with no noise in the training patterns. (See Figure 1 where the value noise level is 0.0.) Notice that solid line and the dotted line cross around where the noise level is 1.2. That is the crossover point where the desired output Jacobians are degraded enough to match the performance of standard backpropagation.

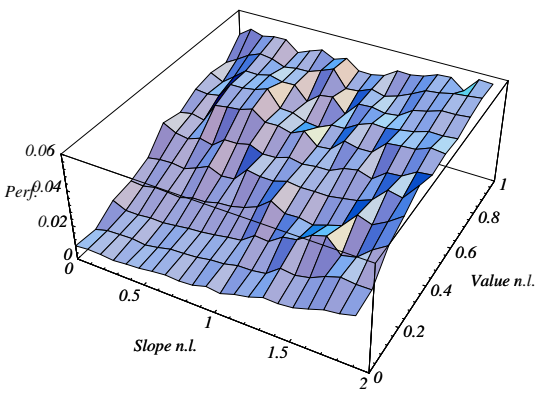

Figure 3: **Performance of EBNN learning (2) :** The performance as defined above of EBNN learning against both the value and slope noise levels is plotted. See the main text for the empirical observations.

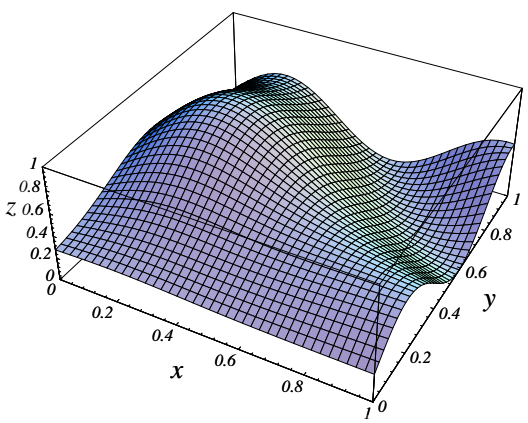

Figure 4: **Target function**