Sensitivity Analysis of Neural Control

Chin-Wang Tao[1], Hung T. Nguyen[2],
J. T. Yao[3], and Vladik Kreinovich[4]

[1]Department of Electrical Engineering
National I-Lan Institute of Technology
260 I-Lan, Taiwan
cwtao@mail.ilantech.edu.tw
[2]Department of Mathematical Sciences
New Mexico State University
Las Cruces, NM 88003, USA
hunguyen@nmsu.edu

[3]Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
jtyao@cs.uregina.ca

[4]Department of Computer Science
University of Texas at El Paso
El Paso, TX 79968, USA
vladik@cs.utep.edu

Abstract

We provide explicit formulas that describe how sensitive the resulting signal of a neural network is to the measurement errors with which we measure the inputs.

What are neural networks. Artificial neural networks (see, e.g., [2]) simulate a highly parallel way the human brain works. In the simplest 3-layer back-propagation neural network, inputs signals $x_1, \ldots, x_n$ first go to K "hidden" neurons. Each of these neurons produces a signal

$$y_k = s_0(w_{k1} \cdot x_1 + \ldots + w_{kn} \cdot x_n + w_{k0}) \quad (1 \le k \le K),$$

where

$$s_0(z) = \frac{1}{1 + \exp(-z)}$$

is an *activation function*. Signals from these neurons are collected at the (linear) *output* neuron, producing the final signal $y = W_1 \cdot y_1 + \ldots + W_K \cdot y_K + W_0$, i.e.,

$$y = \sum_{k=1}^{K} W_k \cdot s_0 \left(\sum_{i=1}^{n} w_{ki} \cdot x_i + w_{k0} \right) + W_0. \qquad (1)$$

Neurons in the hidden layer are called *hidden* because their signals are not directly outputted to the outside world, they are only fed to the output neuron that produces the final result.

Why neural networks. Neural networks are known to be *universal approximators*, i.e., every continuous function $y = f(x_1, \ldots, x_n)$ on a box

$$[-\Delta, \Delta] \times \ldots \times [-\Delta, \Delta],$$

and for every positive real number $\varepsilon > 0$, there exists a function of the type (1) that approximates $f(x_1, \ldots, x_n)$ within a given accuracy ε.

At the same time, they are fast to compute: if we implement all neurons in hardware, then a 3-layer neural network means that no matter how complex the function is, and how many variables it has, it only needs the processing time of two layers to compute the desired value of the function.

We want a neural network to be trained. A typical application of neural networks, e.g., in control, is based on the following idea. Often, we have skilled operators who can control a given plant, but who cannot describe their control in precise terms.

So, what we can do is collect the record of their skillful control, i.e., find out what control $y^{(p)}$ these skilled operators applied for different combinations $x_1^{(p)}, \ldots, x_n^{(p)}$ of input variables, and train a neural network in such a way that it will produce the same control for all given inputs.

How neural networks are trained. The universal approximation result does not tell us *how* to train a neural network, i.e., how to find the values of the weights w_{ki} and W_k that approximate a given function. For this training, one of the most successful algorithms is *back-propagation*, which is, in effect, a gradient descent method for the least square error. Namely, if we want the neural network to produce the output $y^{(p)}$ for given inputs $x_1^{(p)}, \ldots, x_n^{(p)}$, i.e., if we want to minimize the squared difference

$$J = (y^{(p)} - y(x_1^{(p)}, \ldots, x_n^{(p)}))^2, \qquad (2)$$

where $y(x_1,\ldots,x_n)$ denotes the expression (1), then we must update the previously known weights to the new values

$$w_{ki} \to w_{ki} - \lambda \cdot \frac{\partial J}{\partial w_{ki}}; \tag{3}$$

$$W_K \to W_K - \lambda \cdot \frac{\partial J}{\partial W_K}, \tag{4}$$

where $\lambda > 0$ is a step. Back-propagation is, in effect, a fast algorithm for computing the corresponding partial derivatives.

While computation of y from given $x_1,\ldots,x_n$ starts at the hidden neurons and then goes to the output neuron, in the forward direction, the algorithm for computing these derivatives starts with computing the derivatives corresponding to the output neuron, and then moves to computing the derivatives corresponding to the hidden layer, i.e., goes backwards. Thus, this algorithm is called *back-propagation.*

How neural networks are used. In accordance with our description:

- first, we train the neural network to produce exactly the desired values, and
- then, we "freeze" the weights, and use the neural network solely in forward propagation mode.

Problem. The problem that we discuss in this paper is that in real-life control applications, the values x_i of input variables come from measurements, and measurements are never 100% accurate. As a result, the values $\widetilde{x}_i$ that we measure may be slightly different from the actual (unknown) x_i values of the corresponding physical quantities, i.e., the measurement error $\Delta x_i \stackrel{\rm def}{=} \widetilde{x}_i - x_i$ is, in general, different from 0.

How does this uncertainty affect the result of the neural network? In other words, how is the computed value $\widetilde{y} \stackrel{\rm def}{=} y(\widetilde{x}_1,\ldots,\widetilde{x}_n)$ different from the desired value $y \stackrel{\rm def}{=} y(x_1,\ldots,x_n)$? In yet other words, how sensitive is the neural network to the inaccuracy with which we know the inputs?

Measurement errors are usually relatively small. Measurement errors are usually relatively small. So, to find the bounds on

$$\Delta y \stackrel{\rm def}{=} \widetilde{y} - y = y(\widetilde{x}_1,\ldots,\widetilde{x}_n) - y(x_1,\ldots,x_n) =$$

$$y(\widetilde{x}_1,\ldots,\widetilde{x}_n) - y(\widetilde{x}_1 - \Delta x_1,\ldots,\widetilde{x}_n - \Delta x_n), \tag{5}$$

we can expand the dependence (5) in Taylor series and retain only linear terms in this expansion:

$$\Delta y = \frac{\partial y}{\partial x_1} \cdot \Delta x_1 + \ldots + \frac{\partial y}{\partial x_n} \cdot \Delta x_n. \tag{6}$$

Two cases: interval and statistical. The resulting estimate on Δy depends on what we know about the measurement errors Δx_i. In all cases, the manufacturer of the measuring instrument provides us with an upper bound Δ_i on the measurement error: $|\Delta x_i| \le \Delta_i$.

In some situations, this is the only information that we have. In such situations (see, e.g., [4, 5, 6, 7, 8]), the largest possible value of Δy is equal to

$$\Delta = \left|\frac{\partial y}{\partial x_1}\right| \cdot \Delta_1 + \ldots + \left|\frac{\partial y}{\partial x_n}\right| \cdot \Delta_n. \tag{7}$$

In other cases, in addition to the upper bound on the measurement errors, we know the probabilities of different values of these errors. Usually (see, e.g., [10]), the corresponding probability distributions of Δx_i are independent, and each Δx_i is normally distributed with 0 average and known standard deviation σ_i. In this case, the variance $V[y]$, i.e., the mean squared value of Δy, can be computed as follows:

$$V[y] = \left(\frac{\partial y}{\partial x_1}\right)^2 \cdot \sigma_1^2 + \ldots + \left(\frac{\partial y}{\partial x_n}\right)^2 \cdot \sigma_n^2. \tag{8}$$

In both cases, to estimate the effect, we must know the values of the partial derivatives

$$\frac{\partial y}{\partial x_i}. \tag{9}$$

What is known. Several papers (see, e.g., [1, 3]) describe how to compute the desired derivatives (9) on the training stage. On this stage, we know the partial derivatives of J w.r.t. weights, and from these derivatives, we can easily estimate the derivative (9).

Specifically, due to the chain rule, the derivative of J w.r.t w_{k0} is equal to

$$\frac{\partial J}{\partial w_{k0}} = 2 \cdot \Delta y \cdot W_k \cdot s_0' \left(\sum_{i=1}^n w_{ki} \cdot x_i + w_{k0}\right), \tag{10}$$

where we denoted $\Delta y \stackrel{\text{def}}{=} y(x_1^{(p)}, \ldots, x_n^{(p)}) - y^{(p)}$, while

$$\frac{\partial y}{\partial x_i} = \sum_{k=1}^K W_k \cdot s_0' \left(\sum_{i=1}^n w_{ki} \cdot x_i + w_{k0}\right) \cdot w_{ki}. \tag{11}$$

Therefore,

$$\frac{\partial y}{\partial x_i} = \frac{1}{2\Delta y}\sum_{k=1}^{K} \frac{\partial J}{\partial w_{k0}} \cdot w_{ki}. \qquad (12)$$

What we will do. In this paper, we will provide an estimate for the desired partial derivative on the usage stage, when no partial derivatives are known.

Preliminary results of this research were first announced in [11].

Our formula. The resulting formula is as follows:

$$\left|\frac{\partial y}{\partial x_i}\right| \le \frac{1}{4}\cdot \max\left(\sum_k (W_k \cdot w_{ki})^+, \sum_k (W_k \cdot w_{ki})^-\right), \qquad (13)$$

where, for each real number a, $a^+ \overset{\text{def}}{=} \min(a, 0)$ and $a^- \overset{\text{def}}{=} \min(-a, 0)$. In other words, the sum of a^+ is the sum of all positive terms $W_k \cdot w_{ki}$, and the sum of all a^- is the sum of the absolute values of all negative terms.

Proof of correctness. Let us first prove that this formula is indeed correct. Indeed, due to (11), we have

$$\frac{\partial y}{\partial x_i} = \sum_{k=1}^{K} W_k \cdot w_{ki} \cdot s_0'\left(\sum_{i=1}^{n} w_{ki} \cdot x_i + w_{k0}\right). \qquad (14)$$

It is known that $s_0'(z) = s_0(z) \cdot (1 - s_0(z))$. The value $s_0(z)$ goes from 0 to 1, hence $s_0'(z)$ is always non-negative, and its largest value is attained when $s_0(z) = 0.5$; then $s_0'(z) = 0.5 \cdot (1 - 0.5) = 1/4$ (in this case, $z = 0$). So, $s_0'(z) \le 1/4$ for all z.

If the desired partial derivative is positive, then its value cannot exceed the sum of all the positive terms in the expression (14). Since $s_0'(z)$ is always positive, the sign of a term is determined by the product $W_k \cdot w_{ki}$. Thus, if the desired partial derivative is positive, then

$$\left|\frac{\partial y}{\partial x_i}\right| = \frac{\partial y}{\partial x_i} \le \sum_k (W_k \cdot w_{ki})^+ \cdot s_0'(z) \le \frac{1}{4}\cdot \sum_k (W_k \cdot w_{ki})^+. \qquad (15)$$

Similarly, if the desired partial derivative is negative, then its absolute value cannot exceed the sum of absolute values of the negative terms in the sum, i.e.,

$$\left|\frac{\partial y}{\partial x_i}\right| \le \sum_k (W_k \cdot w_{ki})^- \cdot s_0'(z) \le \frac{1}{4}\cdot \sum_k (W_k \cdot w_{ki})^-. \qquad (16)$$

Combining (15) and (16), we conclude that in both cases, the absolute value of the desired partial derivative cannot exceed the largest of these two bounds. In other words, the formula (13) is indeed correct.

Can we get a better estimate? A natural question is: can we get a better estimate? We will show if the number of hidden neurons does not exceed the number of inputs (i.e., $K \le n$), then, in almost all cases, the above estimate cannot be improved.

Specifically, these estimates cannot be improved in the *generic* case, when the corresponding weight vectors $\vec{w}_k \stackrel{\text{def}}{=} (w_{k1}, \ldots, w_{kn})$ are linearly independent.

Let us show that in this case, for every $\varepsilon > 0$, there exist values $x_1, \ldots, x_n$ for which the upper bound is (13) is attained within accuracy ε, i.e., for which

$$\left|\frac{\partial y}{\partial x_i}\right| \ge \frac{1}{4} \cdot \max\left(\sum_k (W_k \cdot w_{ki})^+, \sum_k (W_k \cdot w_{ki})^-\right) - \varepsilon. \quad (17)$$

Without losing generality, let us consider the case when

$$\sum_k (W_k \cdot w_{ki})^+ \ge \sum_k (W_k \cdot w_{ki})^-.$$

In this case, the desired equality (17) takes the equivalent form

$$\frac{\partial y}{\partial x_i} \ge \frac{1}{4} \cdot \sum_k (W_k \cdot w_{ki})^+ - \varepsilon. \quad (18)$$

Let $\mathcal{K}$ be the set of all the indices k for which $W_k \cdot w_{ki} > 0$. Let us fix a large number N and find the values x_i for which:

$$w_{k1} \cdot x_1 + \ldots + w_{kn} \cdot x_n + w_{k0} = 0 \text{ for } k \in \mathcal{K}, \quad (19)$$

$$w_{k1} \cdot x_1 + \ldots + w_{kn} \cdot x_n + w_{k0} = N \text{ for } k \notin \mathcal{K}. \quad (20)$$

Since $K \le n$, and the vectors $\vec{w}_k$ are linearly independent, this system of equations always has a solution, For this solution, the formula (14) leads to:

$$\frac{\partial y}{\partial x_i} = \frac{1}{4} \cdot \sum_k (W_k \cdot w_{ki})^+ - s'(N) \cdot \sum_k (W_k \cdot w_{ki})^-. \quad (21)$$

As $N \to \infty$, we have $s_0'(N) = s_0(N) \cdot (1 - s_0(N)) \to 0$, hence, for large enough N, we have the inequality (18).

Thus, our bound cannot indeed be improved. The statement is proven.

Acknowledgments

This work was supported in part by NASA under cooperative agreement NCC5-209, by the Future Aerospace Science and Technology Program (FAST) Center for Structural Integrity of Aerospace Systems, effort sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-1-0365, by NSF grants EAR-0112968 and EAR-0225670, by the Army Research Laboratories grant DATM-05-02-C-0046, by the Hewlett-Packard equipment grants 89955.1 and 89955.2, by the Personal Interface AccessGrid awarded by the Education, Outreach and Training Partnership for Advanced Computational Infrastructure EOT-PACI, and by the IEEE/ACM SC2003 Minority Serving Institutions Participation Grant.

References

[1] C. Alippi, V. Piuri, and M. Sami, "Sensitivity to errors in Artificial Neural Networks: a behavioral approach", *IEEE Transactions on Circuits and Systems, I: Fundamental Theory and Applications*, 1995, Vol. 42, No. 6, pp. 358–361.

[2] L. Fausett, *Fundamentals of neural networks: Architectures, algorithms, and applications*, Prentice Hall, Englewood Cliffs, NJ, 1994.

[3] S. Hashem, "Sensitivity analysis for feedforward Artificial Neural Networks with differentiable activation functions", *Proceedings of the 1992 International Joint Conferences on Neural Networks, Baltimore, MD*, IEEE Press, 1992, Vol. 1, pp. 419–424.

[4] L. Jaulin, M. Keiffer, O. Didrit, and E. Walter, *Applied Interval Analysis*, Springer-Verlag, Berlin, 2001.

[5] R. B. Kearfott, *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht, 1996.

[6] R. B. Kearfott and V. Kreinovich (eds.), *Applications of Interval Computations*, Kluwer, Dordrecht, 1996.

[7] R. E. Moore, *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.

[8] A. Neumaier, *Introduction to Numerical Analysis*, Cambridge Univ. Press, Cambridge, 2001.

[9] H. T. Nguyen and V. Kreinovich, *Applications of continuous mathematics to computer science*, Kluwer, Dordrecht, 1997.

[10] S. Rabinovich, *Measurement Errors: Theory and Practice*, American Institute of Physics, New York, 1993.

[11] J. T. Yao, "Sensitivity analysis for data mining", *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2003*, Chicago, Illinois, July 24–26, 2003, pp. 420–425.