Interval Computations
No 4, 1993

Neural Networks That Are Not Sensitive To The Imprecision of Hardware Neurons

Ongard Sirisaengtaksin, Vladik Kreinovich*

It has been proved that 3-layer neural networks can approximate any continuous function with any given precision (Hecht-Nielsen, Cybenko, Funahashi, Hornik, Stinchcombe, White).

These theorems mean, in particular, that for a plant whose state can be described by finitely many parameters $x_1, \ldots, x_n$, for an arbitrary control $u = f(x_1, \ldots, x_n)$, and for an arbitrary precision $\varepsilon > 0$, we can implement this control with a given precision using a 3-layer neural network. However, they are based on the assumption that the neurons themselves are ideal. What if they are not?

In the present paper, we prove that such realistic (non-precise) neurons are also universal approximators. Namely, we prove that it is possible for an arbitrary continuous function f, and a real number $\varepsilon > 0$, to produce a design (scheme) of a neural network and the necessary precision δ in such a way that even if all the neurons are built with this precision δ, the in-out characteristics of the resulting network will be ε-close to f.

Нейронные сети, нечувствительные к погрешностям аппаратно реализованных нейронов

О. Сирисенгтаксин, В. Крейнович

Доказано, что трехслойная нейронная сеть может вычислить приближенное значение любой непрерывной функции с любой заданной точностью (Хехт-Нильсен, Цыбенко, Фунахаши, Хорник, Стинчкомб, Уайт).

Этот результат, в частности, означает, что для объекта, состояние которого описывется конечным числом параметров $x_1, \ldots, x_n$, и для произвольно заданной точности $\varepsilon > 0$ управляющее воздействие $u = f(x_1, \ldots, x_n)$ можно реализовать с помощью трехслойной нейронной сети. Эта теорема,

*This work was supported in part by NSF Grant No. CDA-9015006, NSF Research Opportunity Award (for O.S.), NASA Research Grant No. 9-482, and a grant from the Institute for Materials and Manufacturing Management. The authors are thankful to S. Aityan, A. Neumaier, and anonymous referees for valuable comments.

однако, предполагает, что сами нейроны являются идеальными. Как же обстоит дело в обратном случае?

Нами доказано, что такие <реальные> (неточные) нейроны также пригодны для универсальной аппроксимации. Конкретно, мы доказываем, что для произвольной непрерывной функции f и действительного числа $\varepsilon > 0$ возможно построить схему нейронной сети и задать точность δ так, что даже если все нейроны построены с этой точностью δ, выходная характеристика сети будет отличаться от f не более чем на ε.

1 Introduction

Neural networks are often used for control design. For non-linear systems, it is often very difficult to find an explicit analytical expression for a control that satisfies certain given goals. It is even more difficult to design a control that is optimal in some reasonable sense (e.g., uses the minimal amount of fuel, attains its goals in the shortest time, etc). One of the methods that (in many cases) helps to design such a control is to train a neural network in such a way that for a given input $\vec{x}$, its output is close to the ideal control value u.

What are the limits of this methodology? Sometimes this method helps, sometimes it does not. If after, say, 3000 iterations the network is still not appropriately trained, does it mean that it cannot be trained in principle, or that we were not sufficiently patient (and after more iterations, we would have got the desired control)?

Neural networks are universal approximators. An answer to this question was given by several authors who proved that 3-layer neural networks can approximate any continuous function $f(\vec{x})$ with any given precision [2–6]. These results are extremely valuable for control: they show that for a plant whose state can be described by finitely many parameters, for an arbitrary control $u(\vec{x})$, and for an arbitrary precision $\varepsilon > 0$, we can implement this control with a given precision using 3-layer neural networks. In other words, these results mean that, *in principle*, an arbitrary control can be obtained by using neural networks.

These results are based on idealized neurons. Actual hardware neurons are not precise. We said and arbitrary control can be "in principle" obtained by using neural networks, because these results are based on the assumption that we can design neurons with precisely known in-out

characteristics $s(x)$. In reality, however, it is technically impossible to design a hardware device whose in-out characteristics $\tilde{s}(x)$ precisely coincides with a prescribed function $s(x)$. What we can guarantee is as follows: we can fix a precision $\delta > 0$, and the biggest possible signal X, and guarantee that the in-out characteristics $\tilde{s}(x)$ of a manufactured (hardware) neuron belongs to an interval $[s(x) - \delta, s(x) + \delta]$ for all x from $-X$ to X.

Our main result is that with these non-precise ("interval") neurons, it is possible for an arbitrary continuous function f, and a real number $\varepsilon > 0$, to produce a design (scheme) of a neural network and the necessary precision δ in such a way that even if we build all the neurons with this precision δ, the in-out characteristics of the resulting network will be ε-close to f.

This result first appeared in a Technical Report [7]. In this report, we also show how to build designs that are not sensitive to the parameters of the neurons, for plants with distributed parameters (in which the state of the plant is described by a function, and the control takes that function as an input), and for several other control problems.

How this result is related to interval computations. One of the main goals of interval computations is to provide guaranteed results for numerical computations, i.e., to provide the users with intervals that are guaranteed to contain the desired result. To achieve this goal, we must use only hardware for which some accuracy is guaranteed.

Neural networks are inherently parallel and, therefore, fast. Hence, if we want to design a computational device that computes a given function $f(x_1, \ldots, x_n)$ with a given accuracy $\varepsilon > 0$ faster than a standard (sequential) computer, then a hardware neural network is one of the most promising approaches.

The main result of our paper is that neural networks are indeed a universal tool for this problem, i.e., that we can really design a hardware neural network tool for an arbitrary function f, and for an arbitrary accuracy ε.

The structure of the paper is as follows: in Section 2, we present the above-mentioned classical result. In Section 3, we explain its importance for control theory. In Section 4, we describe the design that is not sensitive to the parameters of the neuron. The proof of this result (for reader's convenience) is given in Section 5.

2 Classical result: neural networks are universal approximators for functions

Let us first recall the classical result that neurons are universal approximators for functions.

Definition 1. *Suppose that a monotonic continuous function $s(x) : R \to (0,1)$ is given such that*

$$\lim_{x\to-\infty} s(x) = 0, \ \lim_{x\to\infty} s(x) = 1.$$

We say that a function $f(x_1,\ldots,x_n) : R^n \to R^p$ is representable by a 3-layer neural network if

$$f_l(x_1,\ldots,x_n) = \sum_{k=1}^{K} \beta_{kl} s\Big(\sum_{i=1}^{n} w_{ki}x_i + b_k\Big)$$

for some integer K, and real numbers β_{kl}, w_{ki}, and b_k, where $1 \le k \le K$, $1 \le i \le n$, and $1 \le l \le p$.

Comments.

- This definition describes a network with 3 layers: an input layer, a hidden layer, and an output layer. The input layer consists of n neurons that read n input values $x_1,\ldots,x_n$. The hidden layer consists of K neurons that input x_i and generate the signals $y_k = s\big(\sum_{i=1}^{n} w_{ki}x_i + b_k\big)$, $1 \le k \le K$. Finally, the output layer consists of p linear neurons that combine the signals y_k into the outputs $\sum_k \beta_{kl}y_k$.

- In Definition 1, parameters β_{kl}, w_{ki}, and b_k, can be arbitrary real numbers (of any sign). This definition describes a model of a neuron that is most widely used in artificial neural networks [5]. This model is an oversimplified description of a biological neuron. For more biologically relevant neurons, see, e.g., [1, 8].

  In particular, biological neural networks have more than 3 layers. In this paper, we restrict ourselves to 3-layer networks only. The reason is that in the majority of applications to control and to numerical computations, time is a crucial factor. The more layers we have, the

bigger the processing time. Therefore, 3-layer networks guarantee the fastest data processing.

Other models used in artificial neural networks include *radial* basis functions, for which the output of a neuron is $y = s\left(\sqrt{\sum x_i^2}\right)$ for some function s (see, e.g., [9, 10]).

- It is known [2, 3, 6] that thus defined neural networks are universal approximators for functions. Namely, the following result is true:

Definition 2. *Suppose that $S \subset R^n$ is a set, and $\varepsilon > 0$ is a real number. We say that functions f and g from S to R^p are ε-close on S if*

$$\left|f_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\right| \le \varepsilon$$

for all $(x_1, \ldots, x_n) \in S$ and all l from 1 to p.

Theorem [Hornik, Stinchcombe, White]. *Assume that n and p are positive integers, $S \subset R^n$ is a compact set, $f : S \to R^p$ is continuous, and $\varepsilon > 0$ is a real number. Then, there exists a function $\tilde{f}(x_1, \ldots, x_n)$ that is representable by a 3-layer neural network and that is ε-close to f on S.*

Comment. The number K of neurons in the hidden layer that is necessary to get an ε-approximation to f, depends on f and on ε. In general, the better accuracy we seek (i.e., the smaller $\varepsilon > 0$ we take), the more neurons we need.

3 This classical result is very important for control theory but is not directly applicable to hardware neurons

This result is very important for control. The theorem that we just described is very important for applications of neural networks to control. Namely, suppose that we consider a plant, whose state can be described by finitely many parameters $x_1, \ldots, x_n$, and whose possible controls can be characterized by p control parameters $u_1, \ldots, u_p$. For such a system, to design a control means to find a way to describe proper control values u_l for each state of the plane $(x_1, \ldots, x_n)$. In mathematical terms, a control strategy is a function from R^n to R^p.

In real-life situations, for every physical parameter x_i, there is an a priori bound on its value: velocity cannot take the value that exceed the speed of light, position cannot take the values that exceed the size of the area that we are analyzing, etc. If we denote by X_i the biggest possible value of $|x_i|$, then we must consider only the values $x_i \in [-X_i, X_i]$, and therefore, only the states $(x_1, \ldots, x_n)$ from a compact set

$$S = [-X_1, X_1] \times [-X_2, X_2] \times \cdots \times [-X_n, X_n]$$

are physically possible. Therefore, we can consider only functions from S to R^p.

Every real-life device, no matter how fast it is, produces a continuous change of parameters: we cannot immediately change the position, cannot change the velocity in no time (with infinite acceleration), and even the change in the electric current (that occurs, e.g., when we switch something on or off) is continuous. Therefore, any real (hardware) control is a continuous function from S to R^p.

Since we are talking about a hardware control, and hardware devices cannot be absolutely precise, we cannot guarantee that the actual control will precisely coincide with the function $f : S \to R^p$ that is prescribed by the optimal control theory. However, by imposing more and more strict restrictions on the quality of the hardware parts, we can try to guarantee that the resulting hardware control is as close to the theoretical one as possible.

Hardware neurons are not precise, so this result is not directly applicable to them. In the above approximation theorem, we design a neural network that approximates f if the neurons are precisely described by the function $s(x)$. But if we want to implement the neural network in hardware, then, of course, the characteristics of the actual hardware neurons will be only approximately equal to $s(x)$. In view of that, it is necessary to design an approximating network so that it will provide the desired approximation for actual neurons as well.

4 A design that is not sensitive to the parameters of a neuron

4.1 Motivation: neurons can be only approximately implemented in hardware

In-out characteristic of a neuron can be only approximately implemented in hardware. It is technically impossible to design a hardware device whose in-out characteristics $\tilde{s}(x)$ coincides (precisely coincides) with a prescribed function $s(x)$. What we can expect is that whatever precision $\delta > 0$ we require, it will be possible to create a hardware neuron whose response for all x that do not exceed some level X is $\delta-$close to $s(x)$

$$\left|\tilde{s}(x) - s(x)\right| \le \delta.$$

This can be done by tuning the device for all values x from this interval $(-X, X)$.

An actual hardware neuron can only process signals of limited size ($|x| \le X$ for some X). It is, of course, impossible, to tune for all possible x, for one reason that the ability of the existing testers to generate big signals is limited. Even if we overcome this difficulty by making an improvement each time, then, in finite time, we can still generate only finitely many signals $X_1, \ldots, X_N$, and so there is no way to directly check our hardware neuron for $|x| > \max\left(|X_1|, \ldots, |X_N|\right)$.

We can check a neuron indirectly: by designing a hardware neuron in such a way that any signal $x > X$ simply unleashes a generator that generates an output signal that is "equal" to 1 (in reality, we cannot make it actually equal 1, but we can make it δ-close to 1). Likewise, we can design $-X$ as another threshold so that for $x < -X$, the generated signal is close to 0.

To guarantee the continuity of the resulting piece-wise defined transformation, we must choose δ and X in such a way that $s(-X) \le \delta$ and $s(X) \ge 1 - \delta$. We will call such values δ and X *consistent.*

Thus, although we cannot design a hardware neuron that gives precisely $s(x)$, for every consistent pair of values δ and X, we can (in principle) design a neuron whose output $\tilde{s}(x)$ satisfies the following inequalities: $|\tilde{s}(x)-$

$s(x)| \le \delta$ for $x \in [-X, X]$, $|\tilde{s}(x)| \le \delta$ for $x < -X$, and $|1 - \tilde{s}(x)| \le \delta$ for $x > X$.

In hardware, the values of the coefficients can be implemented only approximately. Likewise, it is impossible to get the precise values of the coefficients β_{kl}, w_{ki}, and b_k, so, we can only guarantee these values with some precision $\delta > 0$.

Now, we are ready to formulate the results. We restrict them to the case when the basic function $s(x)$ is continuous, because, as we have already mentioned, in real life all functions are continuous.

4.2 Definitions

Definition 3. *Assume that $s(x)$ is a monotonic continuous function $s : R \to (0, 1)$ such that*

$$\lim_{x \to -\infty} s(x) = 0, \ \lim_{x \to \infty} s(x) = 1$$

and K, n, and p are positive integers. By a design of a 3-layer neural network (or design for short) we mean a tuple $\mu = \big(\{\beta_{kl}\}, \{w_{ki}\}, \{b_k\}, X, \delta\big)$, where:

- *β_{kl}, w_{ki}, and b_k are real numbers, $1 \le k \le K$, $1 \le i \le n$, $1 \le l \le p$, and*
- *X and δ are positive real numbers such that $s(-X) \le \delta$ and $s(X) \ge 1 - \delta$ (real numbers X and δ that satisfy these inequalities will be called consistent).*

The pair (X, δ) will be called a manufacturing precision of a design μ.

Definition 4. *A function $\tilde{s} : R \to R$ is called an implementation of a neuron with a manufacturing precision (X, δ) if for $x \in [-X, X]$, $|\tilde{s}(x) - s(x)| \le \delta$; for $x < -X$, $|\tilde{s}(x)| \le \delta$; and for $x > X$, $|\tilde{s}(x) - 1| \le \delta$.*

Definition 5. *By implementation of a design $\mu = \big(\{\beta_{kl}\}, \{w_{ki}\}, \{b_k\}, X, \delta\big)$ we mean a function*

$$\tilde{f}_l(x_1, \ldots, x_n) = \sum_{k=1}^{K} \tilde{\beta}_{kl} \tilde{s}_k \Big(\sum_{i=1}^{n} \tilde{w}_{ki} x_i + \tilde{b}_k \Big)$$

where each function $\tilde{s}_k$ is an implementations of a neuron with manufacturing precision (X, δ), and $\tilde{\beta}_{kl}, \tilde{w}_{ki}$, and $\tilde{b}_k$ satisfy the following inequalities:

$$|\tilde{\beta}_{kl} - \beta_{kl}| \le \delta, \quad |\tilde{w}_{ki} - w_{ki}| \le \delta, \quad \text{and} \quad |\tilde{b}_k - b_k| \le \delta.$$

Definition 6. *Suppose that $S \subset R^n$. We say that a function $f : S \to R^p$ is ε-approximated by a design μ if, for every implementation $\tilde{f}$ of this design, $|f_l(x) - \tilde{f}_l(x)| \le \varepsilon$ for all $x \in S$.*

4.3 Main result

Main theorem. *Assume that n and p are positive integers, $S \subset R^n$ is a compact set, $f : S \to R^p$ is continuous, and $\varepsilon > 0$ is a real number. Then, there exists a design μ that ε-approximates $f(x)$.*

5 Proof of the main theorem

1^o. According to the classical approximation theorem, there exists a function g that is representable by a 3-layer neural network, and that is $(\varepsilon/3)$-close to f. Let us take the parameters β_{kl}, w_{ki}, and b_i of this network as parameters of our design. To complete the description of the design, it is necessary to describe X and δ.

2^o. We want to find such X and δ that for every implementation $\tilde{f}$, we have

$$\left|\tilde{f}_l(x_1, \ldots, x_n) - f_l(x_1, \ldots, x_n)\right| \le \varepsilon.$$

We already have the inequality $|f_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)| \le \varepsilon/3$. So, we will be able to prove the desired inequality, if we can guarantee that

$$\left|\tilde{f}_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\right| \le \frac{2}{3}\varepsilon.$$

3^o. The difference between $\tilde{f}$ and g is due to two reasons: neurons are non-ideal, and coefficients of $\tilde{f}$ can be different from the coefficients from g. Let us separate the effects of non-ideal neurons and of non-ideal coefficients.

To perform this separation, let us consider auxiliary functions

$$h_l(x_1,\ldots,x_n)=\sum_{k=1}^{K}\tilde{\beta}_{kl}s\Big(\sum_{i=1}^{n}\tilde{w}_{ki}x_i+\tilde{b}_k\Big)$$

and try to find such X and δ that for manufacturing precision (X,δ), $|\tilde{f}_l(x_1,\ldots,x_n)-h_l(x_1,\ldots,x_n)|\le\varepsilon/3$, and $|h_l(x_1,\ldots,x_n)-g_l(x_1,\ldots,x_n)|\le\varepsilon/3$.

4^o. Let us first find the conditions, under which the first inequality will be guaranteed, i.e., under which $|\tilde{f}_l(x_1,\ldots,x_n)-h_l(x_1,\ldots,x_n)|\le\varepsilon/3$.

4.1^o. The difference between the function

$$\tilde{f}_l(x_1,\ldots,x_n)=\sum_{k=1}^{K}\tilde{\beta}_{kl}\tilde{s}_k\Big(\sum_{i=1}^{n}\tilde{w}_{ki}x_i+\tilde{b}_k\Big)$$

and $h_l(x_1,\ldots,x_n)$ can be represented as

$$\tilde{f}_l(x_1,\ldots,x_n)-h_l(x_1,\ldots,x_n)=\sum_{k=1}^{K}\tilde{\beta}_{kl}\Big[\tilde{s}_k\Big(\sum_{i=1}^{n}\tilde{w}_{ki}x_i+\tilde{b}_k\Big)-s\Big(\sum_{i=1}^{n}\tilde{w}_{ki}x_i+\tilde{b}_k\Big)\Big].$$

Therefore, if we can find an estimate Δ for $|\tilde{s}(x)-s(x)|$ for all x, we can guarantee that

$$\big|\tilde{f}_l(x_1,\ldots,x_n)-h_l(x_1,\ldots,x_n)\big|\le\Big(\sum_{k}|\tilde{\beta}_{kl}|\Big)\Delta.$$

So, let us find such an estimate Δ.

4.2^o. Suppose that $\delta>0$ is given. Then, since $s(x)\to 1$ as $x\to\infty$, we conclude that there exists an X^+ such that for $x>X^+$, we have $|s(x)-1|\le\delta$. Likewise, from $s(x)\to 0$ as $x\to-\infty$, we conclude that there exists a X^- such that for $x<X^-$, we have $|s(x)|\le\delta$. Let us take $X=\max(X^+,|X^-|)$. Then, if $x>X$, then $|s(x)-1|\le\delta$. If $x<-X$, then $|s(x)|\le\delta$. So, the values δ and X are consistent in the sense of Definition 3.

If $\tilde{s}(x)$ is an implementation of a neuron $s(x)$ with a manufacturing precision (X,δ), then, for $x\in[-X,X]$, we have $|\tilde{s}(x)-s(x)|\le\delta$. For $x<-X$, we have $|\tilde{s}(x)|\le\delta$, hence $|\tilde{s}(x)-s(x)|\le|\tilde{s}(x)|+|s(x)|\le 2\delta$. For $x>X$, we have $|1-\tilde{s}(x)|\le\delta$ and $|1-s(x)|\le\delta$, hence

$$\big|\tilde{s}(x)-s(x)\big|=\big|\big(1-\tilde{s}(x)\big)-\big(1-s(x)\big)\big|\le\big|1-\tilde{s}(x)\big|+\big|1-s(x)\big|\le 2\delta$$

In all three cases, $|\tilde{s}(x) - s(x)| \le 2\delta$. So, for this choice of X, we can take $\Delta = 2\delta$.

4.3^o. Since $|\tilde{s}(x) - s(x)| \le 2\delta$, we can conclude that

$$\left|\tilde{f}_l(x_1, \ldots, x_n) - h_l(x_1, \ldots, x_n)\right| \le 2\sum_k |\tilde{\beta}_{kl}|\delta.$$

Now, from $|\tilde{\beta}_{kl} - \beta_{kl}| \le \delta$, we conclude that $|\tilde{\beta}_{kl}| \le |\beta_{kl}| + |\tilde{\beta}_{kl} - \beta_{kl}| \le |\beta_{kl}| + \delta$, therefore,

$$\sum_k |\tilde{\beta}_{kl}| \le \sum_k |\beta_{kl}| + K\delta$$

and

$$\left|\tilde{f}_l(x_1, \ldots, x_n) - h_l(x_1, \ldots, x_n)\right| \le 2\delta\Big(\sum_k |\tilde{\beta}_{kl}| + K\delta\Big) \le 2\delta\Big(\max_l \sum_k |\beta_{kl}| + K\delta\Big).$$

The expression in the right hand side of this inequality tends to 0 as $\delta \to 0$, therefore, there exists a δ_0 such that if $0 \le \delta \le \delta_0$, then

$$2\delta\Big(\sum_k |\beta_{kl}| + K\delta\Big) \le \frac{\varepsilon}{3}.$$

For such δ, $|\tilde{f}_l(x_1, \ldots, x_n) - h_l(x_1, \ldots, x_n)| \le \varepsilon/3$.

4.4^o. So, we will choose δ in such a way that $\delta \le \delta_0$, and X as in 4.2^o.

5^o. Let us now find for what δ the second inequality

$$|h_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)| \le \varepsilon/3$$

is true.

5.1^o. The expression

$$h_l(x_1, \ldots, x_n) = \sum_{k=1}^{K} \tilde{\beta}_{kl} s\Big(\sum_{i=1}^{n} \tilde{w}_{ki} x_i + \tilde{b}_k\Big)$$

can be viewed as a function of x_i, $\tilde{\beta}_{kl}, \tilde{w}_{ki}$ and $\tilde{b}_k$: $h_l(x_1, \ldots, x_m) = G(\nu)$, where we denoted $\nu = \big(\{x_i\}, \{\tilde{\beta}_{kl}\}, \{\tilde{w}_{ki}\}, \{\tilde{b}_k\}\big)$. Since $s(x)$ is continuous, this function G is also continuous.

In particular, for $\nu = \nu^{(0)} = \big(\{x_i\}, \{\beta_{kl}\}, \{w_{ki}\}, \{b_k\}\big)$, we have

$$G(\nu^{(0)}) = g_l(x_1, \ldots, x_n).$$

Therefore, in terms of G, the difference

$$\big|h_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\big|$$

takes the form $|G(\nu) - G(\nu^{(0)})|$. So, we will use the continuity of G to find such δ, for which this difference is $\le \varepsilon/3$.

5.2^o. Let us first prove that the domain of G is compact.

Indeed, since $|\tilde{\beta}_{kl} - \beta_{kl}| \le \delta \le \delta_0$, we conclude that the possible values of $\tilde{\beta}_{kl}$ are all from the interval $[\beta_{kl} - \delta_0, \beta_{kl} + \delta_0]$. Likewise, $\tilde{w}_{ki} \in [w_{ki} - \delta_0, w_{ki} + \delta_0]$ and $\tilde{b}_k \in [b_k - \delta_0, b_k + \delta_0]$. The values of x belong to a compact set S. So, the possible values of x_i, $\tilde{\beta}_{kl}, \tilde{w}_{ki}$, and $\tilde{b}_k$ form a compact set. Let us denote this compact set by K.

On this set K, we can introduce a $\max-$metric as follows: if

$$\nu^{(1)} = \big(\{x_i^{(1)}\}, \{\tilde{\beta}_{kl}^{(1)}\}, \{\tilde{w}_{ki}^{(1)}\}, \{\tilde{b}_k^{(1)}\}\big)$$

and

$$\nu^{(2)} = \big(\{x_i^{(2)}\}, \{\tilde{\beta}_{kl}^{(2)}\}, \{\tilde{w}_{ki}^{(2)}\}, \{\tilde{b}_k^{(2)}\}\big)$$

then

$$d(\nu^{(1)}, \nu^{(2)}) =$$

$$\max(\max_i |x_i^{(1)} - x_i^{(2)}|, \max_{k,l} |\tilde{\beta}_{kl}^{(1)} - \tilde{\beta}_{kl}^{(2)}|, \max_{k,i} |\tilde{w}_{ki}^{(1)} - \tilde{w}_{ki}^{(2)}|, \max_k |\tilde{b}_k^{(1)} - \tilde{b}_k^{(2)}|).$$

5.3^o. So, the function G is continuous on a compact set K. Therefore, G is uniformly continuous on K. This means that for every $\alpha > 0$, there exists a $\beta > 0$ such that $d(\nu^{(1)}, \nu^{(2)}) \le \beta$, then $|G(\nu^{(1)}) - G(\nu^{(2)})| \le \alpha$. In particular, such a β exists for $\alpha = \varepsilon/3$.

Then, for

$$\nu = \big(\{x_i\}, \{\tilde{\beta}_{kl}\}, \{\tilde{w}_{ki}\}, \{\tilde{b}_k\}\big)$$

and

$$\nu^{(0)} = \big(\{x_i\}, \{\beta_{kl}\}, \{w_{ki}\}, \{b_k\}\big)$$

if the manufacturing precision is (X, δ), we have $d(\nu, \nu^{(0)}) \le \delta$. So, if we choose $\delta \le \beta$, then we can guarantee that

$$\big|G(\nu - \nu^{(0)})| = |h_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\big| \le \alpha = \varepsilon/3$$

for all x_i.

6^o. Summarizing: we have found the values δ_0 and $\beta > 0$ such that if $\delta \le \beta$ and $\delta \le \delta_0$, and X is determined as in 4.2^o, then

$$\left|f_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\right| \le \frac{\varepsilon}{3},$$

$$\left|\tilde{f}_l(x_1, \ldots, x_n) - h_l(x_1, \ldots, x_n)\right| \le \frac{\varepsilon}{3},$$

and

$$\left|h_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\right| \le \varepsilon/3.$$

Therefore, if we take $\delta = \min(\delta_0, \beta)$, we conclude that for every implementation $\tilde{f}$,

$$\left|\tilde{f}_l(x_1, \ldots, x_n) - f_l(x_1, \ldots, x_n)\right| \le \left|\tilde{f}_l(x_1, \ldots, x_n) - h_l(x_1, \ldots, x_n)\right| +$$

$$\left|h_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\right| + \left|f_l(x_1, \ldots, x_n) - g_l(x_1, \ldots, x_n)\right| \le \varepsilon. \text{ Q.E.D.}$$

References

[1] Carpenter, G. and Grossberg, S. *Pattern recognition by self-organizing neural network*. MIT Press, 1991.

[2] Cybenko, G. *Approximation by superpositions of a sigmoidal function*. Mathematics of Control, Signals and System **2** (1989), pp. 303–314.

[3] Funahashi, K. *On the approximate realization of continuous mappings by neural networks*. Neural Networks **2** (1989), pp. 183–192.

[4] Hecht-Nielsen, R. *Kolmogorov's mapping neural network existence theorem*. In: "IEEE International Conference on Neural Networks", San Diego, SOS Printing, 1987, pp. 11–14.

[5] Hecht-Nielsen, R. *Neurocomputing*. Addison-Wesley, Reading, MA, 1990.

[6] Hornik, K., Stinchcombe, M., and White, H. *Multilayer feedforward networks are universal approximators*. Neural Networks **2** (1989), pp. 359–366.

[7] Kreinovich, V. and Sirisaengtaksin, O. *3-layer neural networks are universal approximators for functionals and for control strategies*. University of Texas at El Paso, Computer Science Department, Technical Report UTEP–CS–92–27, 1992.

[8] Levine, D. *Introduction to neural and cognitive modeling*. Lawrence Elbaum, 1991.

[9] Poggio, T. and Girosi, F. *Regularization algorithms for learning that are equivalent to multilayer networks*. Science **247** (1990), pp. 978–981.

[10] Powell, M. J. D. *The theory of radial basis function approximation*. In: Light, W. A. (ed.) "Advances in numerical analysis. II", Oxford, Oxford University Press, 1992.

[11] Sprecher, D. A. *Elements of real analysis*. Dover Publ., N.Y., 1987.

O. Sirisaengtaksin
Department of Computer and
Mathematical Sciences
University of Houston-Downtown
Houston, TX 77002
USA

V. Kreinovich
Computer Science Department
University of Texas at El Paso
El Paso, TX 79968
USA