1-2009

Towards Neural-Based Understanding of the Cauchy Deviate Method for Processing Interval and Fuzzy Uncertainty

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Hung T. Nguyen

Comments:
Technical Report: UTEP-CS-09-07a
Published in *Proceedings of the 2009 World Congress of the International Fuzzy Systems Association IFSA*, Lisbon, Portugal, July 20-24, 2009, pp. 1264-1269.

Towards A Neural-Based Understanding of the Cauchy Deviate Method for Processing Interval and Fuzzy Uncertainty

Vladik Kreinovich[1] and Hung T. Nguyen[2]

[1]Department of Computer Science, University in the Texas at El Paso
500 W. University, El Paso, TX 79968, USA, vladik@utep.edu
[2]Department of Mathematical Sciences, New Mexico State University
Las Cruces, NM 88003, USA, hunguyen@nmsu.edu

Abstract— *One of the most efficient techniques for processing interval and fuzzy data is a Monte-Carlo type technique of Cauchy deviates that uses Cauchy distributions. This technique is mathematically valid, but somewhat counterintuitive. In this paper, following the ideas of Paul Werbos, we provide a natural neural network explanation for this technique.*

Keywords— Cauchy deviate method, fuzzy uncertainty, interval uncertainty, Monte-Carlo simulations, neural networks

1 Formulation of the Problem: Cauchy Deviate Method and Need for Intuitive Explanation

1.1 Practical Need for Uncertainty Propagation

In many practical situations, we are interested in the value of a quantity y which is difficult or even impossible to measure directly. To estimate this difficult-to-measure quantity y, we measure or estimate related easier-to-measure quantities $x_1, \ldots, x_n$ which are related to the desired quantity y by a known relation $y = f(x_1, \ldots, x_n)$. Then, we apply the relation f to the estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$ for x_i and produce an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y.

In the simplest cases, the relation $f(x_1, \ldots, x_n)$ may be an explicit expression: e.g., if we know the current x_1 and the resistance x_2, then we can measure the voltage y by using Ohm's law $y = x_1 \cdot x_2$. In many practical situations, the relation between x_i and y is much more complicated: the corresponding algorithm $f(x_1, \ldots, x_n)$ is not an explicit expression, but a complex algorithm for solving an appropriate non-linear equation (or system of equations).

Estimates are never absolutely accurate:

- measurements are never absolutely precise, and
- expert estimates can only provide approximate values of the directly measured quantities $x_1, \ldots, x_n$.

In both cases, the resulting estimates $\widetilde{x}_i$ are, in general, different from the actual (unknown) values x_i. Due to these estimation errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$, even if the relation $f(x_1, \ldots, x_n)$ is exact, the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ is different from the actual value $y = f(x_1, \ldots, x_n)$: $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y \neq 0$.

(In many situations, when the relation $f(x_1, \ldots, x_n)$ is only known approximately, there is an additional source of the approximation error in y caused by the uncertainty in knowing this relation.)

It is therefore desirable to find out how the uncertainty Δx_i in estimating x_i affects the uncertainty Δy in the desired quantity, i.e., how the uncertainties Δx_i *propagate* via the algorithm $f(x_1, \ldots, x_n)$.

1.2 Propagation of Probabilistic Uncertainty

Often, we know the probabilities of different values of Δx_i. For example, in many cases, we know that the approximation errors Δx_i are independent normally distributed with zero mean and known standard deviations σ_i; see, e.g., [16].

In this case, we can use known statistical techniques to estimate the resulting uncertainty Δy in y. For example, since we know the probability distributions, we can simulate them in the computer, i.e., use the Monte-Carlo simulation techniques to get a sample population $\Delta y^{(1)}, \ldots, \Delta y^{(N)}$ of the corresponding errors Δy. Based on this sample, we can then estimate the desired statistical characteristics of the desired approximation error Δy.

1.3 Propagation of Interval Uncertainty

In many other practical situations, we do not know these probabilities, we only know the upper bounds Δ_i on the (absolute values of) the corresponding measurement errors Δx_i: $|\Delta x_i| \leq \Delta$.

In this case, based on the known approximation $\widetilde{x}_i$, we can conclude that the actual (unknown) value of i-th auxiliary quantity x_i can take any value from the interval

$$\mathbf{x}_i = [\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]. \tag{1}$$

To find the resulting uncertainty in y, we must therefore find the range $\mathbf{y} = [\underline{y}, \overline{y}]$ of possible values of y when $x_i \in \mathbf{x}_i$:

$$\mathbf{y} = f(\mathbf{x}_1, \ldots, \mathbf{x}_n) \stackrel{\text{def}}{=}$$

$$\{f(x_1, \ldots, x_n) \,|\, x_1 \in \mathbf{x}_1, \ldots, x_n \in \mathbf{x}_n\}. \tag{2}$$

Computations of this range under interval uncertainty is called *interval computations*; see, e.g., [4, 5].

The corresponding computational problems are, in general, NP-hard [9]. Crudely speaking, this means that, in general, such problems require a large amount of computation time – and that therefore faster methods are needed.

1.4 Propagation of Fuzzy Uncertainty

In many practical situations, the estimates $\widetilde{x}_i$ come from experts. Experts often describe the inaccuracy of their estimates

in terms of imprecise words from natural language, such as "approximately 0.1", etc. A natural way to formalize such words is to use special techniques developed for formalizing this type of estimates – specifically, the technique of fuzzy logic; see, e.g., [6, 15].

In this technique, for each possible value of $x_i \in \mathbf{x}_i$, we describe the degree $\mu_i(x_i)$ to which this value is possible. For each degree of certainty α, we can determine the set of values of x_i that are possible with at least this degree of certainty – the *α-cut* $\mathbf{x}_i(\alpha) = \{x \,|\, \mu(x) \geq \alpha\}$ of the original fuzzy set. Vice versa, if we know α-cuts for every α, then, for each object x, we can determine the degree of possibility that x belongs to the original fuzzy set [3, 6, 12, 13, 15]. A fuzzy set can be thus viewed as a nested family of its (interval) α-cuts.

We already know how to propagate interval uncertainty. Thus, to propagate this fuzzy uncertainty, we can therefore consider, for each α, the fuzzy set y with the α-cuts

$$\mathbf{y}(\alpha) = f(\mathbf{x}_1(\alpha), \ldots, \mathbf{x}_1(\alpha)); \tag{3}$$

see, e.g., [3, 6, 12, 13, 15]. So, from the computational viewpoint, the problem of propagating fuzzy uncertainty can be reduced to several interval propagation problems.

1.5 Need for Faster Algorithms for Uncertainty Propagation

Summarizing the above analysis, we can conclude that in principle, we need to consider three basic types of uncertainty propagation: situations when we propagate probabilistic, interval, and fuzzy uncertainty. It is also possible that some quantities are represented by fuzzy sets, while others may be represented by probabilities.

For probabilistic uncertainty, there exist reasonable efficient uncertainty propagation algorithms such as Monte-Carlo simulations. In contrast, the problems of propagating interval and fuzzy uncertainty are, in general, computationally difficult. It is therefore desirable to design faster algorithms for propagating interval and fuzzy uncertainty.

Once such methods are developed, we can then use these methods to propagate interval and fuzzy uncertainty components, and Monte-Carlo simulations to propagate the probabilistic uncertainty.

The computational problem of propagating fuzzy uncertainty can be naturally reduced to the problem of propagating interval uncertainty. Because of this reduction, in the following text, we will mainly concentrate on faster algorithms for propagating interval uncertainty.

1.6 Linearization Situations: Description

Due to the approximation errors $\Delta x_i = \widetilde{x}_i - x_i$, the unknown (actual) values $x_i = \widetilde{x}_i - \Delta x_i$ of the input quantities x_i are, in general, different from the approximate estimates $\widetilde{x}_i$. In many practical situations, the approximation errors Δx_i are small – e.g., when the approximations are obtained by reasonably accurate measurements. In such situations, we can ignore terms which are quadratic (and of higher order) in Δx_i.

1.7 Linearization Situations: Analysis

In the above situations, we can expand the expression for

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

$$f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \Delta x_1, \ldots, \widetilde{x}_n - \Delta x_n) \tag{4}$$

in Taylor series in Δx_i and keep only the linear terms in this expansion. In this case, we get

$$\Delta y = c_1 \cdot \Delta x_1 + \ldots + c_n \cdot \Delta x_n, \tag{5}$$

where we denoted

$$c_i \stackrel{\text{def}}{=} \frac{\partial f}{\partial x_i}(\widetilde{x}_1, \ldots, \widetilde{x}_n).$$

For a linear function, the largest possible value of Δy is obtained when each of the variables $\Delta x_i \in [-\Delta_i, \Delta_i]$ attains:

- either its largest value Δ_i (when $c_i \geq 0$)
- or its smallest value $-\Delta_i$ (when $c_i < 0$).

In both cases, the largest possible value of the corresponding term in Δy is equal to $|c_i| \cdot \Delta_i$. Thus, the largest possible value of Δy is equal to

$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n. \tag{6}$$

Similarly, the smallest possible value of Δy is obtained when each of the variables $\Delta x_i \in [-\Delta_i, \Delta_i]$ attains

- either its smallest value $-\Delta_i$ (when $c_i \geq 0$)
- or its largest value Δ_i (when $c_i < 0$).

In both cases, the smallest possible value of the corresponding term in Δy is equal to $-|c_i| \cdot \Delta_i$. Thus, the smallest possible value of Δy is equal to

$$-\Delta = -|c_1| \cdot \Delta_1 - \ldots - |c_n| \cdot \Delta_n. \tag{7}$$

Can we transform these natural formulas into an algorithm? Due to the linearization assumption, we can estimate each partial derivative c_i as

$$c_i \approx \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h_i} \tag{8}$$

for some small values h_i. So, we arrive at the following algorithm.

1.8 Linearization Situations: Algorithm

To compute the range $\mathbf{y}$ of y, we do the following.

- First, we apply the algorithm f to the original estimates $\widetilde{x}_1, \ldots, \widetilde{x}_n$, resulting in the value $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$.
- Second, for all i from 1 to n, we compute $f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n)$ for some small h_i and then compute

$$c_i = \frac{f(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h_i, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n) - \widetilde{y}}{h_i}. \tag{9}$$

- Finally, we compute

$$\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n$$

and the desired range $\mathbf{y} = [\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

1.9 Linearization Situations: Computational Complexity

The main computation time is spent on calling the time-consuming algorithm f. In the above uncertainty propagation algorithm, after one call to f to compute $\widetilde{y}$, we need n calls to f to compute the corresponding partial derivatives c_i and then, we can estimate the desired uncertainty Δ in y by using the above simple formula.

Overall, we thus need $n+1$ calls to the algorithm f.

1.10 Cauchy Deviate Method

For large n, we can further reduce the number of calls to f if we use a special technique of Cauchy-based Monte-Carlo simulations, which enables us to use a fixed number of calls to f (≈ 200) for all possible values n; see, e.g., [7, 8].

1.11 Mathematics Behind the Cauchy Method

In our simulations, we use *Cauchy distribution* – i.e., probability distributions with the probability density

$$\rho(z) = \frac{\Delta}{\pi \cdot (z^2 + \Delta^2)}; \tag{10}$$

the value Δ is called the *(scale) parameter* of this distribution.

Cauchy distribution has the following property that we will use: if $z_1, \ldots, z_n$ are independent random variables, and each of z_i is distributed according to the Cauchy law with parameter Δ_i, then their linear combination

$$z = c_1 \cdot z_1 + \ldots + c_n \cdot z_n \tag{11}$$

is also distributed according to a Cauchy law, with a scale parameter $\Delta = |c_1| \cdot \Delta_1 + \ldots + |c_n| \cdot \Delta_n$.

Therefore, if we take random variables δ_i which are Cauchy distributed with parameters Δ_i, then the value

$$\delta \stackrel{\text{def}}{=} f(\widetilde{x}_1, \ldots, \widetilde{x}_n) - f(\widetilde{x}_1 - \delta_1, \ldots, \widetilde{x}_n - \delta_n) =$$

$$c_1 \cdot \delta_1 + \ldots + c_n \cdot \delta_n \tag{12}$$

is Cauchy distributed with the desired parameter

$$\Delta = \sum_{i=1}^{n} |c_i| \cdot \Delta_i. \tag{13}$$

So, repeating this experiment N times, we get N values $\delta^{(1)}, \ldots, \delta^{(N)}$ which are Cauchy distributed with the unknown parameter, and from them we can estimate Δ.

The bigger N, the better estimates we get.

1.12 Cauchy Method: Towards Implementation

To implement this idea, we must answer the following two questions:

- how to simulate the Cauchy distribution; and
- how to estimate the parameter Δ of this distribution from a finite sample.

Simulation can be based on the functional transformation of uniformly distributed sample values:

$$\delta_i = \Delta_i \cdot \tan(\pi \cdot (r_i - 0.5)), \tag{14}$$

where r_i is uniformly distributed on the interval $[0, 1]$.

In order to estimate Δ, we can apply the Maximum Likelihood Method

$$\rho(\delta^{(1)}) \cdot \rho(\delta^{(2)}) \cdot \ldots \cdot \rho(\delta^{(N)}) \to \max, \tag{15}$$

where $\rho(z)$ is a Cauchy distribution density with the unknown Δ. When we substitute the above-given formula for $\rho(z)$ and equate the derivative of the product with respect to Δ to 0 (since it is a maximum), we get an equation

$$\frac{1}{1 + \left(\dfrac{\delta^{(1)}}{\Delta}\right)^2} + \ldots + \frac{1}{1 + \left(\dfrac{\delta^{(N)}}{\Delta}\right)^2} = \frac{N}{2}. \tag{16}$$

The left-hand side of (16) is an increasing function that is equal to $0 (< N/2)$ for $\Delta = 0$ and $> N/2$ for $\Delta = \max\left|\delta^{(k)}\right|$; therefore the solution to the equation (16) can be found by applying a bisection method to the interval $\left[0, \max\left|\delta^{(k)}\right|\right]$.

It is important to mention that we assumed that the function f is reasonably linear within the box

$$[\widetilde{x}_1 - \Delta_1, \widetilde{x}_1 + \Delta_1] \times \ldots \times [\widetilde{x}_n - \Delta_n, \widetilde{x}_n + \Delta_n]. \tag{17}$$

However, the simulated values δ_i may be outside the box. When we get such values, we do not use the function f for them, we use a normalized function that is equal to f within the box, and that is extended linearly for all other values (we will see, in the description of an algorithm, how this is done).

As a result, we arrive at the following algorithm.

1.13 Cauchy Deviates Method: Algorithm

- Apply f to the results of direct measurements:

$$\widetilde{y} := f(\widetilde{x}_1, \ldots, \widetilde{x}_n); \tag{18}$$

- For $k = 1, 2, \ldots, N$, repeat the following:
  - use the standard random number generator to compute n numbers $r_i^{(k)}$, $i = 1, 2, \ldots, n$, that are uniformly distributed on the interval $[0, 1]$;
  - compute Cauchy distributed values

$$c_i^{(k)} := \tan(\pi \cdot (r_i^{(k)} - 0.5)); \tag{19}$$

  - compute the largest value of $|c_i^{(k)}|$ so that we will be able to normalize the simulated measurement errors and apply f to the values that are within the box of possible values: $K := \max_i |c_i^{(k)}|$;
  - compute the simulated measurement errors

$$\delta_i^{(k)} := \Delta_i \cdot c_i^{(k)} / K; \tag{20}$$

  - compute the simulated "actual values"

$$x_i^{(k)} := \widetilde{x}_i - \delta_i^{(k)}; \tag{21}$$

  - apply the program f to the simulated "actual values" and compute the simulated error of the indirect measurement:

$$\delta^{(k)} := K \cdot \left(\widetilde{y} - f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)\right); \tag{22}$$

- Compute Δ by applying the bisection method to solve the equation (16).

Comment. To avoid confusion, we should emphasize that, in contrast to the Monte-Carlo solution for the probabilistic case, the use of Cauchy distribution in the interval case is a computational trick and *not* a truthful simulation of the actual measurement error Δx_i: indeed, we know that the actual value of Δx_i is always inside the interval $[-\Delta_i, \Delta_i]$, but a Cauchy distributed random attains values outside this interval as well.

1.14 Cauchy Deviate Method: Need for Intuitive Explanation

The above Cauchy deviate method is one of the most efficient techniques for processing interval and fuzzy data. However, this method has a serious drawback: while the corresponding technique is mathematically valid, it is somewhat counterintuitive – we want to analyze errors which are located *instead* a given interval $[-\Delta, \Delta]$, but this analysis use Cauchy simulated errors which are located, with a high probability, *outside* this interval.

It is therefore desirable to come up with an intuitive explanation for this technique. In this paper, we show that such an explanation can be obtained from neural networks. (For a general introduction to neural networks, see, e.g., [2, 11].)

2 Solution: Neural Explanation

2.1 Werbos's Idea: Use Neurons

Our explanation comes from the idea promoted by Paul Werbos, the author of the backpropagation algorithm for training neural networks. Traditionally, neural networks are used to simulate a deterministic dependence; Paul Werbos suggested that the same neural networks can be used to describe stochastic dependencies as well – if as one of the inputs, we take a standard random number r uniformly distributed on the interval $[0, 1]$; see, e.g., [18] and references therein.

In view of this idea, as a natural probability distribution, we can take the result of applying a neural network to this random number. The simplest case is when we have a single neuron. In this case, we apply the activation (input-output) function $f(y)$ corresponding to this neuron to the random number r.

So, let us see what will happen if we apply a neuron to the standard random number and get a value $f(r)$.

2.2 What is the Activation Function of a Neuron: Reminder

To answer the above question, let us recall what are the optimal choices of an activation function of a neuron. This problem was analyzed in detail in [14]; see also [10].

2.3 We Must Choose a Family of Functions, Not a Single Function

We talk about choosing f, but the expression for $f(y)$ will change if we change the units in which we measure all the signals (input, output and intermediate), so in mathematical terms, it is better to speak about choosing a *family* of functions f.

It is reasonable to suggest that if an f belongs to this family, then this family must contain $k \cdot f$ for positive real numbers k. This corresponds to changing units.

Also, it must contain $f + c$, where c is a constant. This is equivalent to adding a constant bias and therefore does not change the abilities of the resulting network.

Since we are talking about non-linear phenomena, we can also assume that some non-linear "rescaling" transformations $x \to g(x)$ are also applicable, i.e., the family must include the composition $g(f(y))$ for each of functions f.

This family must not be too big, therefore, it must be determined by finitely many parameters and should ideally be obtained from one function $f(y)$ by applying all these transformations. Without loss of generality, we can assume that this set of transformations is closed under composition and under inverse, i.e., if $z \to g_1(z)$ and $z \to g_2(z)$ are possible transformations, then $z \to g_1(g_2(z))$ and $z \to g_1^{-1}(z)$ are possible transformations, where by g_1^{-1} we denoted an inverse function $g_1^{-1}(z) = w$ if and only if $g_1(w) = z$. In mathematical terms this means that these transformations form a *group*, and therefore a family is obtained by applying to some function $f(y)$ all transformations from some finite-dimensional transformation group G that includes all linear transformations (and maybe some non-linear ones).

All these transformations correspond to appropriate "rescalings". Rescaling is something that is smoothly changing the initial scale. This means that if we have two different transformations, there must be a smooth transition between them. In mathematical terms, the existence of this continuous transition is expressed by saying that the group is *connected*, and the fact that both the transformations and the transitions are smooth is expressed by saying that this is a *Lie group*).

2.4 Which Family is the Best?

Among all such families, we want to choose the best one. In formalizing what "the best" means we follow the general idea described in [14].

The criteria to choose may be computational simplicity, efficiency of training, or something else. In mathematical optimization problems, numeric criteria are most frequently used, when to every family we assign some value expressing its performance, and choose a family for which this value is maximal. However, it is not necessary to restrict ourselves to such numeric criteria only. For example, if we have several different families that have the same training ability A, we can choose between them the one that has the minimal computational complexity C. In this case, the actual criterion that we use to compare two families is not numeric, but more complicated: *a family F_1 is better than the family F_2 if and only if either $A(F_1) > A(F_2)$ or $A(F_1) = A(F_2)$ and $C(F_1) < C(F_2)$*. A criterion can be even more complicated. What a criterion *must* do is to allow us for every pair of families to tell whether the first family is better with respect to this criterion (we'll denote it by $F_1 > F_2$), or the second is better ($F_1 < F_2$) or these families have the same quality in the sense of this criterion (we'll denote it by $F_1 \sim F_2$). Of course, it is necessary to demand that these choices be consistent, e.g., if $F_1 > F_2$ and $F_2 > F_3$ then $F_1 > F_3$.

Another natural demand is that this criterion must choose a *unique* optimal family (i.e., a family that is better with respect to this criterion than any other family). The reason for this demand is very simple. If a criterion does not choose any family at all, then it is of no use. If several different families are "the

best" according to this criterion, then we still have a problem to choose among those "best". Therefore, we need some additional criterion for that choice. For example, if several families turn out to have the same training ability, we can choose among them a family with minimal computational complexity. So what we actually do in this case is abandon that criterion for which there were several "best" families, and consider a new "composite" criterion instead: F_1 is better than F_2 according to this new criterion if either it was better according to the old criterion or according to the old criterion they had the same quality and F_1 is better than F_2 according to the additional criterion. In other words, if a criterion does not allow us to choose a unique best family it means that this criterion is not ultimate; we have to modify it until we come to a final criterion that will have that property.

The next natural condition that the criterion must satisfy is connected with the following. Suppose that instead of a neuron with the transformation function $f(y)$ we consider a neuron with a function $\bar{f}(y) = f(y + a)$, where a is a constant. This new neuron can be easily simulated by the old ones: namely, the output of this new neuron is $\bar{f}(y) = f(y + a)$, so it is equivalent to an old neuron with an additional constant input a. Likewise, the old neuron is equivalent to the new neuron with an additional constant input $-a$. Therefore, the networks that are formed by these new neurons have precisely the same abilities as those that are built from the old ones.

We cannot claim that the new neurons have the same quality as the old ones, because adding a can increase computational complexity and thus slightly worsen the overall quality. But it is natural to demand that adding a does not change the *relative* quality of the neurons, i.e., if a family $\{f(y)\}$ is better that a family of $\{g(y)\}$, then for every a the family $\{f(y+a)\}$ must be still better than the family $\{g(y + a)\}$.

Now, we are ready for the formal definitions.

2.5 Definitions

By a *transformation* we mean a smooth (differentiable) function from real numbers into real numbers. By an *appropriate transformation group* G we mean a finite-dimensional connected Lie group of transformations. By a *family* of functions we mean the set of functions that is obtained from a smooth (everywhere defined) non-constant function $f(y)$ by applying all the transformations from some appropriate transformation group G. Let us denote the set of all the families by F.

A pair of relations $(>, \sim)$ is called *consistent* if it satisfies the following conditions: (1) if $a > b$ and $b > c$ then $a > c$; (2) $a \sim a$; (3) if $a \sim b$ then $b \sim a$; (4) if $a \sim b$ and $b \sim c$ then $a \sim c$; (5) if $a > b$ and $b \sim c$ then $a > c$; (6) if $a \sim b$ and $b > c$ then $a > c$; (7) if $a > b$ then $b > a$ or $a \sim b$ are impossible.

Assume a set A is given. Its elements will be called *alternatives*. By an *optimality criterion* we mean a consistent pair $(>, \sim)$ of relations on the set A of all alternatives. If $a > b$, we say that a is *better* than b; if $a \sim b$, we say that the alternatives a and b are *equivalent* with respect to this criterion. We say that an alternative a is *optimal* (or *best*) with respect to a criterion $(>, \sim)$ if for every other alternative b either $a > b$ or $a \sim b$.

We say that a criterion is *final* if there exists an optimal alternative, and this optimal alternative is unique.

In the present section we consider optimality criteria on the set F of all families.

By the *result of adding* a to a function $f(y)$ we mean a function $\bar{f}(y) = f(y + a)$. By the *result of adding* a to a family F we mean the set of the functions that are obtained from $f \in F$ by adding a. This result will be denoted by $F+a$. We say that an optimality criterion on F is *shift-invariant* if for every two families F and G and for every number a, the following two conditions are true:

i) if F is better than G in the sense of this criterion (i.e., $F > G$), then $F + a > G + a$;

ii) if F is equivalent to G in the sense of this criterion (i.e., $F \sim G$), then $F + a \sim G + a$.

2.6 Main Result

As we have already remarked, the demands that the optimality criterion is final and shift-invariant are quite reasonable. The only problem with them is that at first glance they may seem rather weak. However, they are not, as the following theorem shows:

Theorem. *If a family F is optimal in the sense of some optimality criterion that is final and shift-invariant, then every function f from F has the form $a + b \cdot s_0(K \cdot y + l)$ for some a, b, K and l, where $s_0(y)$ is either a linear function, or a fractional-linear function, or $s_0(y) = \exp(y)$, or the logistic (sigmoid) function $s_0(y) = 1/(1 + \exp(-y))$, or $s_0(y) = \tan(y)$.*

Comment. The logistic function is indeed the most popular activation function for actual neural networks, but others are also used. For our purpose, we will use the tangent function. As we have mentioned earlier, the application of the tangent function to the standard random number r indeed leads to the desired Cauchy distribution.

2.7 Proof: Main Idea

The idea of this proof is as follows: first we prove that the appropriate transformation group consists of fractionally-linear functions (in Part 1), then we prove that the optimal family is shift-invariant (in Part 2), and from that in Part 3 we conclude that any function f from F satisfies some functional equations, whose solutions are known.

2.8 Proof: Part 1

By an *appropriate group* we meant a connected finite-dimensional Lie group of transformations of the set of real numbers R onto itself that contains all linear transformations. Norbert Wiener asked [19] to classify such groups for an n-dimensional space with arbitrary n, and this classification was obtained in [17]. In our case (when $n = 1$) the only possible groups are the group of all linear transformations and the group of all fractionally-linear transformations $x \to (a \cdot x + b)/(c \cdot x + d)$. In both cases the group consists only of fractionally linear transformations.

2.9 Proof: Part 2

Let us now prove that the optimal family F_{opt} exists and is *shift-invariant* in the sense that $F_{opt} = F_{opt} + a$ for all real numbers a. Indeed, we assumed that the optimality criterion

is final, therefore there exists a unique optimal family F_{opt}. Let's now prove that this optimal family is shift-invariant.

The fact that F_{opt} is optimal means that for every other F, either $F_{opt} > F$ or $F_{opt} \sim F$. If $F_{opt} \sim F$ for some $F \neq F_{opt}$, then from the definition of the optimality criterion we can easily deduce that F is also optimal, which contradicts the fact that there is only one optimal family. So for every F either $F_{opt} > F$ or $F_{opt} = F$.

Take an arbitrary a and let $F = F_{opt} + a$. If $F_{opt} > F = F_{opt} + a$, then from the invariance of the optimality criterion (condition ii) we conclude that $F_{opt} - a > F_{opt}$, and that conclusion contradicts the choice of F_{opt} as the optimal family. So $F_{opt} > F = F_{opt} + a$ is impossible, and therefore $F_{opt} = F = F_{opt} + a$, i.e., the optimal family is really shift-invariant.

2.10 Proof: Part 3

Let us now deduce the actual form of the functions f from the optimal family. If $f(y)$ is such a function, then the result $f(y + a)$ of adding a to this function f belongs to $F + a$, and so, due to 2., it belongs to F. But all the functions from f can be obtained from each other by fractionally linear transformations, so $f(y + a) = (A + B \cdot f(y))/(C + D \cdot f(y))$ for some A, B, C and D. So we arrive at a functional equation for f. Let us reduce this equation to a one with a known solution. For that purpose, let us use the fact that fractionally linear transformations are projective transformations of a line, and for such transformations the cross ratio is preserved ([1], Section 2.3), i.e., if $g(y) = (A + B \cdot f(y))/(C + D \cdot f(y))$, then

$$\frac{g(y_1) - g(y_3)}{g(y_2) - g(y_3)} \cdot \frac{g(y_2) - g(y_4)}{g(y_1) - g(y_4)} = \frac{f(y_1) - f(y_3)}{f(y_2) - f(y_3)} \cdot \frac{f(y_2) - f(y_4)}{f(y_1) - f(y_4)} \tag{23}$$

for all y_i. In our case this is true for $g(y) = f(y+a)$, therefore for all a the following equality is true:

$$\frac{f(y_1 + a) - f(y_3 + a)}{f(y_2 + a) - f(y_3 + a)} \cdot \frac{f(y_2 + a) - f(y_4 + a)}{f(y_1 + a) - f(y_4 + a)} = \frac{f(y_1) - f(y_3)}{f(y_2) - f(y_3)} \cdot \frac{f(y_2) - f(y_4)}{f(y_1) - f(y_4)}. \tag{24}$$

The most general continuous solutions of this functional equation are given by Theorem 2.3.2 from [1]: either f is fractionally linear, or $f(y) = (a+b\cdot\tan(k\cdot y))/(c+d\cdot\tan(k\cdot y))$ for some a, b, c, d, or $f(y) = (a+b\cdot\tanh(k\cdot y))(c+d\cdot\tanh(k\cdot y))$, where

$$\tanh(z) \stackrel{\text{def}}{=} \frac{\sinh(z)}{\cosh(z)}, \ \sinh(z) \stackrel{\text{def}}{=} \frac{\exp(z) - \exp(-z))}{2},$$

$$\cosh(z) \stackrel{\text{def}}{=} \frac{\exp(z) + \exp(-z)}{2}. \tag{25}$$

The $\tanh(z)$ expression is equivalent to the logistic function.

The theorem is proven.

Acknowledgment

This work was supported in part by NSF grant HRD-0734825 and by Grant 1 T36 GM078000-01 from the National Institutes of Health. The authors are thankful to Paul Werbos for valuable discussions, and to the anonymous referees for valuable suggestions.

References

[1] J. Aczel, *Lectures on functional equations and their applications,* Dover, New York, 2006.

[2] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York, 2006.

[3] D. Dubois and H. Prade, Operations on fuzzy numbers, *International Journal of Systems Science*, 1978, Vol. 9, pp. 613–626.

[4] Interval computations website http://www.cs.utep.edu/interval-comp

[5] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control and Robotics*, Springer-Verlag, London, 2001.

[6] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*, Prentice Hall, New Jersey, 1995.

[7] V. Kreinovich, J. Beck, C. Ferregut, A. Sanchez, G. R. Keller, M. Averill, and S. A. Starks, "Monte-Carlo-type techniques for processing interval uncertainty, and their potential engineering applications", *Reliable Computing*, 2007, Vol. 13, No. 1, pp. 25–69.

[8] V. Kreinovich and S. Ferson, "A New Cauchy-Based Black-Box Technique for Uncertainty in Risk Analysis", *Reliability Engineering and Systems Safety*, 2004, Vol. 85, No. 1–3, pp. 267–279.

[9] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[10] V. Kreinovich and C. Quintana. "Neural networks: what non- linearity to choose?," *Proceedings of the 4th University of New Brunswick Artificial Intelligence Workshop*, Fredericton, N.B., Canada, 1991, pp. 627–637.

[11] J. Lawrence, *Introduction to Neural Networks*, California Scientific Software Press, 1994.

[12] R. E. Moore and W. Lodwick, Interval Analysis and Fuzzy Set Theory, *Fuzzy Sets and Systems*, 2003, Vol. 135, No. 1, pp. 5–9.

[13] H. T. Nguyen and V. Kreinovich, Nested Intervals and Sets: Concepts, Relations to Fuzzy Sets, and Applications, In: R. B. Kearfott and V. Kreinovich, eds., *Applications of Interval Computations*, Kluwer, Dordrecht, 1996, pp. 245–290.

[14] H. T. Nguyen and V. Kreinovich, *Applications of Continuous Mathematics to Computer Science*, Kluwer, Dordrecht, 1997.

[15] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, CRC Press, Boca Raton, Florida, 2006.

[16] S. Rabinovich, *Measurement Errors and Uncertainties: Theory and Practice*, American Institute of Physics, New York, NY, 2005.

[17] I. M. Singer and S. Sternberg, "Infinite groups of Lie and Cartan, Part 1", *Journal d'Analyse Mathematique*, 1965, Vol. XV, pp. 1–113.

[18] P. J. Werbos, "Intelligence in the brain: A theory of how it works and how to build it", *Neural Networks*, 2009, to appear.

[19] N. Wiener, *Cybernetics, or Control and Communication in the animal and the machine*, MIT Press, Cambridge, MA, 1962.