2-1-2022

Data Processing under Fuzzy Uncertainty: Towards More Efficient Algorithm

Hung T. Nguyen
New Mexico State University, hunguyen@nmsu.edu

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

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

Comments:

Technical Report: UTEP-CS-22-25

Data Processing under Fuzzy Uncertainty: Towards More Efficient Algorithms

1st Hung T. Nguyen
Department of Mathematical Sciences
New Mexico State Univesity
Las Cruces, New Mexico, USA, and
Faculty of Economics
Chiang Mai University
Chiang Mai, Thailand
hunguyen@nmsu.edu

2nd Olga Kosheleva
Department of Teacher Education
University of Texas at El Paso
El Paso, Texas, USA
olgak@utep.edu

3rd Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
El Paso, Texas, USA
vladik@utep.edu

***Abstract*—In many practical situations, we need to process data under fuzzy uncertainty: we have fuzzy information about the algorithm's input, and we want to find the resulting information about the algorithm's output. It is known that this problem can be reduced to computing the range of the algorithm over alpha-cuts of the input. Since the fuzzy degrees are usually known with accuracy at best 0.1, it is sufficient to repeat this range-computing procedure for 11 values alpha = 0, 0.1, ..., 1.0. However, a straightforward application of this idea requires 11 times longer computation time than each range estimation – and for complex algorithms, each range computation is already time-consuming. In this paper, we show that when all inputs are of the same time, we can compute all the desired ranges much faster.**

***Index Terms*—fuzzy data processing, computing the range, interval computations, central form, Cauchy deviates method**

I. Formulation of the Problem

A. Need for Fuzzy Data Processing

Often, we are interested in a quantity y that is difficult to estimate directly. In many such cases, we know a relation $y = f(x_1, \ldots, x_n)$ between this quantity y and auxiliary quantities x_i for which we do have expert estimates.

Usually these expert estimates are formulated in terms of imprecise ("fuzzy") words from a natural language like "small". A natural way to describe such "fuzzy" estimates is to use fuzzy techniques; see, e.g., [1], [3], [8], [11]–[13].

Based on this description, we need to gauge the resulting uncertainty in y.

B. Enter Zadeh's Extension Principle

In fuzzy technique, the information about each input x_i is described by assigning, to each real number X_i, the degree $\mu_i(X_i) \in [0, 1]$ to which, according to the expert, this number is a possible value of x_i. The corresponding function $\mu_i(X_i)$ is known as a *membership function*.

Based on this information, we need to find a similar membership function $\mu(Y)$ for the quantity $y = f(x_1, \ldots, x_n)$ that describes, for each real number Y, the degree to which this number is a possible value of y.

Intuitively, a number Y is a possible value of the quantity y if and only if there exist values $X_1, \ldots, X_n$:

- which are possible values of the corresponding inputs and
- for which $Y = f(X_1, \ldots, X_n)$.

We know the degrees $\mu_i(X_i)$ to which each X_i is a possible value of x_i. So:

- if we use the simplest way to describing "and" and "or" in fuzzy techniques – as min and max – and
- take into account that "there exist" is nothing else by an infinite "or',

we conclude that

$$\mu(Y) =$$
$$\max\{\min(\mu_1(X_1), \ldots, \mu_n(X_n)) : f(X_1, \ldots, X_n) = Y\}.$$

This formula was first introduced by Zadeh and is thus known as *Zadeh's extension principle*.

C. What If the Dependence Between x_i and y Is Only Approximately Known?

In our analysis, we assumed that the dependence $y = f(x_1, \ldots, x_n)$ is *exact*. In many practical situations, however, we only know an *approximate* dependence, i.e., we know that $y \approx f(x_1, \ldots, x_n)$, and we only have fuzzy information about the inaccuracy $m \overset{\text{def}}{=} y - f(x_1, \ldots, x_n)$.

In this case, we actually have the exact dependence $y = f(x_1, \ldots, x_n) + m$ on $n + 1$ inputs $x_1, \ldots, x_n, m$ (i.e., we can view m as x_{n+1}). For each of these $n + 1$ inputs, we know the corresponding membership function. Thus, from the computational viewpoint, this more realistic formulation can be reduced the previously considered case.

In view of this reduction, in the following text, we will assume that the dependence between y and x_i is exactly known.

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), and HRD-1834620 and HRD-2034030 (CAHSI Includes), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

D. How to Perform Fuzzy Data Processing

A known way to perform the corresponding computations is to take into account that for each $\alpha \in [0,1]$, the α-cut

$$\mathbf{y}(\alpha) \stackrel{\text{def}}{=} \{Y : \mu(Y) \ge \alpha\}$$

of y is equal to the range of the function $f(x_1,\ldots,x_n)$ on α-cuts

$$\mathbf{x}_i(\alpha) \stackrel{\text{def}}{=} \{X_i : \mu_i(X_i) \ge \alpha\}$$

of the inputs x_i:

$$\mathbf{y}(\alpha) = \{f(x_1,\ldots,x_n) : x_i \in \mathbf{x}_i(\alpha)\};$$

see, e.g., [10].

Techniques for computing such ranges are known as *interval computation*; see, e.g., [2], [7], [9]. In general, interval computation is NP-hard [6], meaning that for large n, *exact* computation of the range is not always feasible. However, there are efficient *approximate* interval techniques.

E. How Many α-Cuts Do We Need?

Theoretically, there are infinite many possible values $\alpha \in [0,1]$. However, since the fuzzy degrees are usually known with accuracy at best 0.1, it is sufficient to repeat this range-computing procedure for 11 values $\alpha = 0, 0.1, \ldots, 1.0$.

So, to find the corresponding α-cuts for y, we can simply apply interval techniques 11 times.

F. What If Some Information Comes from Measurements?

In this case, for the corresponding input x_i, we know the interval of possible values, and the interval – like every other crips set – is, of course, a particular case of a fuzzy set. So, this case can also be handled the same way as before.

G. Problem: The Current Procedure May Take Too Long

For complex algorithms – and many data processing algorithms are very complex – each range computation is already time-consuming. The need to repeat this computation 11 times increases the computation time by an order of magnitude and can, thus, make the computations not practical.

For example, an accurate prediction of tomorrow's weather may takes several hours on a high-performance computer. If we need to repeat these computations 11 times to find a good description of the accuracy of the prediction results, this will take more than a day – and this makes no sense, since by then we will already observe tomorrow's weather.

It is therefore desirable to speed up computations.

H. What We Do in This Paper

In this paper, we show how to speed up the corresponding computations. We consider two cases:

- the main case, when any approximate method will work, and
- an important auxiliary case when we need to have a guaranteed bound.

For each case, we propose a way to speed up the corresponding computations.

II. Case of Linearization

A. We need to compute an interval range

For usual membership functions, α-cuts are intervals. The data processing algorithms are usually continuous, so the desired range is also an interval. Thus, for each α, we need to compute the range

$$[\underline{y}, \overline{y}] = \{f(x_1,\ldots,x_n) : x_i \in [\underline{x}_i, \overline{x}_i]\}$$

of the given function $f(x_1,\ldots,x_n)$ on given intervals $[\underline{x}_i, \overline{x}_i]$.

As we have mentioned, in general, the exact computation of such a range is NP-hard. Som we can, at best, get an approximate estimate for this range.

B. Cased When Linearlization Is Possible

In many practical situations, the estimation error is relatively small, i.e., the interval width $\overline{x}_i - \underline{x}_i$ is much smaller than the actual values x_i from this interval – e.g., 10% or 20% of this value. Thus, the difference $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$ between the midpoint $\widetilde{x}_i \stackrel{\text{def}}{=} (\underline{x}_i + \overline{x}_i)/2$ and a possible value $x_i \in [\underline{x}_i, \overline{x}_i]$ is also small. This difference is bounded by the interval's half-width $\Delta_i \stackrel{\text{def}}{=} (\overline{x}_i - \underline{x}_i)/2$, so $\Delta x_i \in [-\Delta_i, \Delta_i]$.

In this case, terms quadratic in such small differences are much smaller than linear terms: e.g., square of 10% is 1% which is much smaller than 10%. Thus, we can *linearize* the problem, i.e.:

- expand the difference $\widetilde{y} - y$, where $\widetilde{y} \stackrel{\text{def}}{=} f(\widetilde{x}_1,\ldots,\widetilde{x}_n)$ is the result of processing midpoints, in Taylor series in terms of the small differences Δx_i, and
- keep only linear terms in this expansion.

As a result, we get:

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

$$\sum_{i=1}^n c_i \cdot \Delta x_i, \text{ where } c_i = \frac{\partial f}{\partial x_i}.$$

For a linear function Δy, its largest possible value is attained when Δx_i attains:

- its largest possible value Δ_i when $c_i \ge 0$ and
- its smallest possible value $-\Delta_i$ when $c_i \le 0$.

The resulting largest value Δ of the difference $\Delta y \stackrel{\text{def}}{=} \widetilde{y} - y$ is equal to $\Delta = \sum_{i=1}^n |c_i| \cdot \Delta_i$. Similarly, one can show that the smallest possible value of Δy is $-\Delta$. So, the resulting interval range of possible values of y is $[\widetilde{y} - \Delta, \widetilde{y} + \Delta]$.

C. How This Problem Is Solved Now

To use the above formula for Δ, we need to know the values of all the partial derivatives c_i. For small n, we can feasibly compute all these values by the usual numerical differentiation techniques, e.g., as

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

for some small h_i.

For complex data processing algorithms, however, we often have thousands of inputs, so computing all partial derivatives would take too long. In this case, it is possible to use Cauchy deviate Monte-Carlo method [5], which is based on the fact that if the variables Δx_i are Cauchy distributed with parameters Δ_i, then the linear combination $\Delta y = \sum c_i \cdot \Delta x_i$ is also Cauchy distributed, with parameter $\Delta = \sum |c_i| \cdot \Delta_i$.

In this method – in contrast to numerical differentiation – the number of calls to the algorithm f does not depend on the number of variables n. This number of calls depends only on the desired accuracy and remains constant when n grows.

D. Limitations of The Current Approach

If we follow this algorithm, then, to compute the α-cut for all eleven values of α, we need to call the Cauchy method eleven times.

Can we do it faster? It turns out that we can – in frequent situations when all membership functions are of the same type.

E. What If All Membership Functions Are of the Same Type: Description of the Situation

Often, all membership functions are of the same type: e.g.:

- all are symmetric triangular, or
- all are Gaussian.

In general, this means that all these membership functions $\mu_i(X_i)$ are obtained from some standard membership function $\mu_0(X)$ by some scaling $X \mapsto s \cdot X$ (with $s > 0$) and shift $X \mapsto X + c$, i.e., $\mu_i(X_i) = \mu_0(s_i \cdot X_i + c_i)$ for some s_i and c_i.

For each membership function, there is some "most probable" value – e.g., the midpoint of the 1-cut, i.e., of the set of all the values for which the degree of possibility is 1.

If for the original membership function this value is not 0, we can appropriately shift this standard function, and get a new standard function for which this point is 0. Shifting the standard membership function does not change the class of all membership functions obtained from the standard one by shifts and scalings. Thus, without losing generality, we can safely assume that for the standard function, the "most probable" value is 0.

In this case, the α-cuts for x_i are determined by the α-cuts $[\ell(\alpha), r(\alpha)]$ of the standard membership function $\mu_0(X)$. Indeed, here, $\mu_i(X_i) \ge \alpha$ is equivalent to $\mu_0(s_i \cdot X_i + c_i) \ge \alpha$, i.e., to $s_i \cdot X_i + c_i \in [\ell(\alpha), r(\alpha)]$, or, equivalently, to

$$\ell(\alpha) \le s_i \cdot X_i + c_i \le r(\alpha).$$

Subtracting c_i from all three sides of this inequality and dividing all three sides by $s_i > 0$, we conclude that the condition $\mu_i(X_i) \ge \alpha$ is equivalent to

$$(1/s_i) \cdot \ell(\alpha) + (c_i/s_i) \le X_i \le (1/s_i) \cdot r(\alpha) + (c_i/s_i),$$

i.e., to

$$a_i \cdot \ell(\alpha) + b_i \le X_i \le a_i \cdot \ell(\alpha) + b_i,$$

where we denoted $a_i \overset{\text{def}}{=} 1/s_i$ and $b_i \overset{\text{def}}{=} c_i/s_i$. Thus, the α-cut $\mathbf{x}_i(\alpha)$ of the i-th input has the form

$$\mathbf{x}_i(\alpha) = [a_i \cdot \ell(\alpha) + b_i, a_i \cdot r(\ell) + b_i].$$

In particular, for each of these functions, the most probable value is $a_i \cdot 0 + b_i = b_i$.

How can we speed up computations in this case?

F. What If All Membership Functions Are of the Same Type: Analysis of the Problem

In the linearized case, once we know the value $\widetilde{y} = f(b_1, \ldots, b_n)$ corresponding to the most probable values b_i, for the difference $\Delta y \overset{\text{def}}{=} \widetilde{y} - f(x_1, \ldots, x_n)$, we get the expression $\Delta y = \sum\limits_{i=1}^{n} c_i \cdot \Delta x_i$, where $\Delta x_i \overset{\text{def}}{=} b_i - x_i$. When

$$x_i \in [a_i \cdot \ell(\alpha) + b_i, a_i \cdot r(\ell) + b_i],$$

then the difference $\Delta x_i = b_i - x_i$ belongs to the interval

$$[\Delta_i^-(\alpha), \Delta_i^+(\alpha)] = [-a_i \cdot r(\alpha), -a_i \cdot \ell(\alpha)].$$

Similarly to the above derivation of the linearization case, to compute the range $[\Delta^-(\alpha), \Delta^+(\alpha)]$ of the linear function $\sum c_i \cdot \Delta x_i$ when $\Delta x_i \in [\Delta_i^-(\alpha), \Delta_i^+(\alpha)]$, we can represent each input interval as

$$\left[\widetilde{\Delta}_i(\alpha) - \Delta_i(\alpha), \widetilde{\Delta}_i(\alpha) + \Delta_i(\alpha)\right],$$

where

$$\widetilde{\Delta}_i(\alpha) = \frac{\Delta_i^-(\alpha) + \Delta_i^+(\alpha)}{2} = -a_i \cdot \frac{\ell(\alpha) + r(\alpha)}{2}$$

and

$$\Delta_i(\alpha) = \frac{\Delta_i^+(\alpha) - \Delta_i^-(\alpha)}{2} = a_i \cdot \frac{r(\alpha) - \ell(\alpha)}{2}.$$

In this case, we have

$$[\Delta^-(\alpha), \Delta^+(\alpha)] = [\widetilde{\Delta}(\alpha) - \Delta(\alpha), \widetilde{\Delta}(\alpha) + \Delta(\alpha)],$$

where

$$\widetilde{\Delta}(\alpha) = \sum_{i=1}^{n} c_i \cdot \widetilde{\Delta}_i(\alpha) = \sum_{i=1}^{n} c_i \cdot \left[-a_i \cdot \frac{\ell(\alpha) + r(\alpha)}{2}\right] = A \cdot \frac{\ell(\alpha) + r(\alpha)}{2}.$$

Here, we denoted

$$A \overset{\text{def}}{=} -\sum_{i=1}^{n} c_i \cdot a_i = \frac{f(b_1 - a_1 \cdot h, \ldots, b_n - a_n \cdot h) - \widetilde{y}}{h} \quad (1)$$

for some small h.

Similarly,

$$\Delta(\alpha) = \frac{r(\alpha) - \ell(\alpha)}{2} \cdot B,$$

where we denoted

$$B \overset{\text{def}}{=} \sum_{i=1}^{n} |c_i| \cdot a_i. \quad (2)$$

The expression (2) can be computed by using the Cauchy deviate method.

Thus, we arrive at the following algorithm.

G. Resulting Algorithm

We are given:

- a function $f(x_1, \ldots, x_n)$,
- values $\ell(\alpha)$ and $r(\alpha)$ describing the shape of the common membership function, and
- values a_i and b_i describing specific membership functions for each inputs x_i.

First:

- we compute the values $\widetilde{y} = f(b_1, \ldots, b_n)$ and (1) simply by calling the algorithm f, and
- we compute the expression (2) by using the Cauchy deviate method.

Then, for each α, we compute the desired range $\mathbf{y}(\alpha)$ as

$$\mathbf{y}(\alpha) =$$
$$\left[\widetilde{y} + A \cdot \frac{\ell(\alpha) + r(\alpha)}{2} - B \cdot \frac{r(\alpha) - \ell(\alpha)}{2},\right.$$
$$\left.\widetilde{y} + A \cdot \frac{\ell(\alpha) + r(\alpha)}{2} + B \cdot \frac{r(\alpha) - \ell(\alpha)}{2}\right].$$

H. How Faster Is This Algorithm?

- In the currently used approach, we need to call the Cauchy method 11 times.
- In the new algorithm, we only call the Cauchy method once.

I. What About More General Families of Membership Functions?

So far, we considered the case when the α-cuts of all membership functions are described by a linear expression with two parameters depending on i. It is possible to consider more general families of membership functions, in which the corresponding linear expression depends on $p \geq 3$ parameters – e.g., the families:

- of all possible (not necessarily symmetric) triangular functions, or
- of all possible trapezoid functions.

In this case, similar formulas show that we can compute the desired range by calling Cauchy method $p - 1$ times. For $p < 12$, this is still better than the original method.

This idea also takes care of the case when:

- some membership functions belong to one family (e.g., triangular) and
- some belong to another family (e.g, Gaussian).

In this case, we can view both linear expressions as a particular case of a general linear formula with more parameters. Thus, we can use the same idea as in the previous paragraph.

III. Case of Proper Interval Techniques

A. Why Go Beyond Linearization

Linearlization methods are approximate, they may slightly overestimate or underestimate the desired range. In some applications, it is very important to make sure that the actual value y does not exceed a certain threshold; e.g.:

- that the nuclear power station does not go into the critical regime,
- that the pollution level of a chemical plant does not exceed the allowed concentrations,
- that the predicted strong wind does not exceed the threshold at which damage to power lines is possible, etc.

To have this guarantee, it is important to makes sure that all possible values y are included in our interval estimate, i.e., that this interval estimate contains (encloses) the actual range. Such estimates are known as *enclosures* for the actual range.

B. Naive Interval Computation: Description and Limitations

A naive – not very good – method for computing the enclosure is to use so-called *naive interval computation*.

This method is based ion the fact that for the cases when data processing consists of a single arithmetic operation, we can explicitly compute the range of the resulting value:

$$[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2];$$
$$[\underline{x}_1, \overline{x}_1] - [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2];$$
$$[\underline{x}_1, \overline{x}_1] \cdot [\underline{x}_2, \overline{x}_2] = [\min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2),$$
$$\max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2);$$
$$1/[\underline{x}_1, \overline{x}_1] = [1/\overline{x}_1, 1/\underline{x}_1] \text{ if } 0 \notin [\underline{x}_1, \overline{x}_1];$$
$$[\underline{x}_1, \overline{x}_1]/[\underline{x}_2, \overline{x}_2] = [\underline{x}_1, \overline{x}_1] \cdot (1/[\underline{x}_2, \overline{x}_2]).$$

These formulas are known as formulas of *interval arithmetic*.

In a computer, any algorithm is implemented as a sequence of arithmetic operations. If we replace each arithmetic operation with the corresponding operation of interval arithmetic, we get an enclosure.

For example, when a computer computes the value of a function $f(x) = x \cdot (1 - x)$, it:

- first computes the difference $r = 1 - x$, and
- then computes the product $x \cdot r$.

So, to find an enclosure for the range of this function on the interval $[0, 1]$, we can:

- first apply interval subtraction to find the range for r as
$$[1, 1] - [0, 1] = [1 - 1, 1 - 0] = [0, 1],$$
and
- then apply interval multiplication to compute
$$[0, 1] \cdot [0, 1] =$$
$$[\min(0 \cdot 0, 0 \cdot 1, 1 \cdot 0, 1 \cdot 1), \max(0 \cdot 0, 0 \cdot 1, 1 \cdot 0, 1 \cdot 1)] =$$
$$[0, 1].$$

The resulting range $[0, 1]$ is clearly an enclosure for the actual range $[0, 0.25]$, but a very crude one.

Comment. In general, naive interval computation replaces each arithmetic operation with numbers with at least two operations with numbers – and is, thus, at least twice longer than computing a single value of the function $f(x_1, \ldots, x_n)$.

C. More Adequate – Even Asymptotically Optimal – Interval Techniques

Interval computation packages computed much narrower (and thus, more practically useful) enclosures. One of the main ideas is to use centered form.

In this technique, on each input interval $[\underline{x}_i, \overline{x}_i]$, we select a representative value $\widetilde{x}_i$. This could be a midpoint, this could be a different point from this interval.

Then, each possible value $x_i \in [\underline{x}_i, \overline{x}_i]$ can be represented as $\widetilde{x}_i - \Delta x_i$, where

$$\Delta x_i \in [\Delta_i^-, \Delta_i^+] \stackrel{\text{def}}{=} [\widetilde{x}_i - \overline{x}_i, \widetilde{x}_i - \underline{x}_i].$$

The centered form technique is based on the Intermediate Value Theorem, according to which for each combination of values $\Delta x_i \in [\Delta_i^-, \Delta_i^+]$, there exist values ξ_{ij} from the same intervals $[\Delta_i^-, \Delta_i^+]$ for which

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

$$\sum_{i=1}^n \frac{\partial f}{\partial x_i}_{|x_j = \widetilde{x}_j + \xi_{ij}} \cdot \Delta x_i.$$

We know that, since $\xi_{ij} \in [\Delta_i^-, \Delta_i^+]$, each partial derivative value belongs to the range of this partial derivative on this interval, and thus, belongs the enclosure $\mathbf{D}_i([\Delta_i^-, \Delta_i^+])$. This enclosure can be computed, e.g., by naive interval computation.

Also, we know that $\Delta x_i \in [\Delta_i^-, \Delta_i^+]$. Thus, we conclude that

$$\Delta y \in \mathbf{D} \stackrel{\text{def}}{=} \sum_{i=1}^n \mathbf{D}_i([\Delta_i^-, \Delta_i^+]) \cdot [\Delta_i^-, \Delta_i^+].$$

The right-hand side of this formula is what is called the centered form.

It is known that this feasible-to-compute formula is asymptotically the most accurate, in the following sense:

- for some constant C, it provides the $C \cdot h^2$ accuracy in estimating the range, where h is largest width of the input intervals, while
- estimating the range with higher accuracy $c \cdot h^2$ is NP-hard for sufficiently small c; see, e.g., [4].

D. Fuzzy Case: Limitations

If we apply this technique for each α, i.e., if for each α, we compute

$$\mathbf{D}(\alpha) = \sum_{i=1}^n \mathbf{D}_i([\Delta_i^-(\alpha), \Delta_i^+(\alpha)]) \cdot [\Delta_i^-(\alpha), \Delta_i^+(\alpha)],$$

then we need to compute n enclosures for partial derivatives 11 times – and, as we mentioned, computing an enclosure is rather time-consuming.

E. What We Propose: Idea and the Resulting Algorithm

Instead of computing the enclosures for the partial derivatives for every α, why not use the fact that all α-cuts are all subsets of the α-cut corresponding to $\alpha = 0$ – and thus, all the values of the partial derivative are contained in the enclosure corresponding to $\alpha = 0$.

So, we can compute such enclosures only once – and then use the formula

$$\mathbf{D}(\alpha) = \sum_{i=1}^n \mathbf{D}_i([\Delta_i^-(0), \Delta_i^+(0)]) \cdot [\Delta_i^-(\alpha), \Delta_i^+(\alpha)].$$

Good news is that the resulting expression – while slightly less accurate – is still asymptotically optimal.

References

[1] R. Belohlavek, J. W. Dauben, and G. J. Klir, Fuzzy Logic and Mathematics: A Historical Perspective. New York: Oxford University Press, 2017.

[2] L. Jaulin, M. Kiefer, O. Didrit, and E. Walter, Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control, and Robotics. London: Springer, 2001.

[3] G. Klir and B. Yuan, Fuzzy Sets and Fuzzy Logic. Upper Saddle River, New Jersey: Prentice Hall, 1995.

[4] V. Kreinovich, "Range estimation is NP-hard For ε^2 accuracy and feasible for $\varepsilon^{2-\delta}$", Reliable Computing, vol. 8, No. 6, pp. 481–491, 2002.

[5] V. Kreinovich and S. Ferson, "A new Cauchy-based black-box technique for uncertainty in risk analysis", Reliability Engineering and Systems Safety, vol. 85, No. 1-3, pp. 267–279, 2004.

[6] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, Computational Complexity and Feasibility of Data Processing and Interval Computations. Dordrecht: Kluwer, 1998.

[7] G. Mayer, Interval Analysis and Automatic Result Verification. Berlin: de Gruyter, 2017.

[8] J. M. Mendel, Uncertain Rule-Based Fuzzy Systems: Introduction and New Directions. Cham, Switzerland: Springer, 2017.

[9] R. E. Moore, R. B. Kearfott, and M. J. Cloud, Introduction to Interval Analysis. Philadelphia: SIAM, 2009.

[10] H. T. Nguyen, "A note on the extension principle for fuzzy sets", Journal of Mathematical Analysis and Applications, vol. 64, pp. 369–380, 1978.

[11] H. T. Nguyen, C. L. Walker, and E. A. Walker, A First Course in Fuzzy Logic. Boca Raton, Florida: Chapman and Hall/CRC, 2019.

[12] V. Novák, I. Perfilieva, and J. Močkoř, Mathematical Principles of Fuzzy Logic. Boston, Dordrecht: Kluwer, 1999.

[13] L. A. Zadeh, "Fuzzy sets", Information and Control, 1965, vol. 8, pp. 338–353, 1965.