10-2020

Data Analytics Beyond Traditional Probabilistic Approach to Uncertainty

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

Comments:

Technical Report: UTEP-CS-20-107

Data Analytics Beyond Traditional Probabilistic Approach to Uncertainty

Vladik Kreinovich
Department of Computer Science
University of Texas at El Paso
500 W. University
El Paso, TX 79968, USA
vladik@utep.edu

Abstract

Data for processing mostly comes from measurements, and measurements are never absolutely accurate: there is always the "measurement error" – the difference between the measurement result and the actual (unknown) value of the measured quantity. In many applications, it is important to find out how these measurement errors affect the accuracy of the result of data processing. Traditional data processing techniques implicitly assume that we know the probability distributions. In many practical situations, however, we only have partial information about these distributions. In some cases, all we know is the upper bound on the absolute value of the measurement error. In other cases, data comes not from measurements but from expert estimates. In this paper, we explain how to estimate the accuracy of the results of data processing in all these situations. We tried to explain not only *what* methods can be used, but also *why* these methods have been proposed and have been successfully used. We hope that this overview will be helpful both to users solving practical problems and to researchers interested in extending and improving the existing techniques.

Keywords: data processing, uncertainty, interval uncertainty, imprecise probabilities, fuzzy uncertainty, machine learning, deep learning

1 Why go beyond traditional probabilistic approach to uncertainty

Why do we need data processing. What do we want? We want to know what will happen in the future – and we want to decide what to do to make the future the most beneficial. For example, we want to be able to predict tomorrow's weather – and we want to find the best way to regulate the temperature and humidity in our offices. We want to know where a spaceship will be several weeks from now – and if needed, what is the best way to correct its trajectory.

To be able to predict the future, we need to know the current state of the world, and we need to know the equations that describe how this state will change. For example, to predict the trajectory of a spaceship:

- we need to know its current location,
- we need to know its distance from the Sun and from other celestial bodies, and
- we can then use Newton's equations to predict its future locations.

The current state of the world can be described by the values of different quantities. Some of these quantities we can measure directly – e.g., we can measure the current temperature and humidity at different locations. Some of the quantities we need to measure indirectly – e.g., we can compute the distance from the spaceship to different celestial bodies if we know the location of the spaceship and the locations of the bodies.

What we get from direct measurements is what is called *data*, and when we process this data – whether to compute the current values of different quantities, to predict their future values, or to find the parameters of the influence that will lead to better future values – this is what is called *data processing.*

Uncertainty is ubiquitous. Most of the data comes from measurements, and measurements are never 100% accurate: the result $\widetilde{x}$ of measuring a quantity x is, in general, different from the actual (unknown) value x of this quantity; see, e.g., [13]. Why is uncertainty ubiquitous?

- There are usually physical reasons for the resulting uncertainty: there are some factors that we cannot take into account – what we call noise – that affect the measurement results. For example, in many accurate measurements, the thermal noise provides a random – thus unpredictable – effect.
- There are also mathematical reasons for uncertainty: each measuring instrument can have only finitely many possible results – e.g., it produced finitely many bits, and with b bits, we can form only 2^b possible sequences. On the other hand, the actual value can be any real number – at least any real number from some interval – and there are infinitely many real numbers on each interval. So, inevitably, the measurement result should be, in general, different from the actual value.

It is important to take uncertainty into account when processing data. In general, in data processing, we apply some algorithm to the input data and get the result. Let us denote the inputs by $x_1, \ldots, x_n$, and the algorithm by f. In these terms, the result y of data processing has the form $y = f(x_1, \ldots, x_n)$.

We apply this data processing algorithm to the measurement results $\widetilde{x}_1, \ldots, \widetilde{x}_n$ and get the estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y.

Even in the ideal case, when the algorithm f describes the exact relation between the actual values of the quantities $x_1, \ldots, x_n$, and y, since the measurement results $\widetilde{x}_i$ are, in general, different from the actual value x_i, our estimate

$$\widetilde{y} = f\left(\widetilde{x}_1, \ldots, \widetilde{x}_n\right)$$

is, in general, different from the desired value

$$y = f(x_1, \ldots, x_n).$$

In many practical situations, it is important to know how accurate it is. For example, if we want a spaceship to get close to an asteroid to take photos, and we estimated the future distance to be 100 m, then:

- if it is 100 ± 10, this is great, but
- if this is 100 ± 200, then maybe the spaceship will crash into the asteroid.

Similarly, if we estimate that an oil field contains 300 million tons of oil, then:

- if it is 300 ± 100, we can start exploring right away, but
- if it is 300 ± 400, maybe there is no oil at all, so we better perform additional measurements before we invest a lot of money into exploration.

Traditional probabilistic approach to uncertainty. How can we determine the measurement accuracy? A natural idea is to *calibrate* each measuring instrument, i.e., to compare, several times, the results of measuring the same quantity:

- by this instrument and
- by a much more accurate instrument (known as a *standard instrument*), so much more accurate that we can ignore its own measurement uncertainty and safely assume that this standard instrument produced accurate results.

Based on several observed differences between the values measured by the calibrated measuring instrument and the standard one:

- we can find possible values of the measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$, and
- we can also find the frequency of different possible values Δx.

In other words, we can find the *probability distribution* of the measurement error Δx.

This possibility underlies the traditional probabilistic approach to uncertainty, when we assume that we know the probability distributions for all the input measurement errors $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$.

Need to go beyond the probabilistic approach: interval uncertainty. While in principle, in most practical situations, it is possible to determine the probability distributions of all measurement errors, there are two important classes of situations when this is not done.

The first class is state-of-the-art measurements. Calibration of a measuring instrument requires that we have another instrument which is much more accurate. But what is the instrument that we use is the best we have? For example, it would be nice to have a five times more accurate telescope to float near the Hubble telescope – but the Hubble telescope is the best we have.

In such situations, we do not know the probability distribution of the measurement error. At best, we know the upper bound Δ on the absolute value $|\Delta x|$ of the measurement error:

$$|\Delta x| \le \Delta.$$

And this upper bound we *must* have: otherwise, if there is no upper bound and thus, for each measurement result, the actual value can be arbitrarily large or arbitrarily small, this is not a measurement, this is a wild guess.

Once we know the measurement result $\widetilde{x}$ and the upper bound Δ on the absolute value of the measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$, the only information that we can conclude about the actual (unknown) value x is that this value is contained in an interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$. Because of this, such uncertainty is known as *interval uncertainty*; see, e.g., [3, 8, 10, 13].

Another class of situations when we do not know probabilities – much more common and much more frequent class – is when we perform measurements during manufacturing. In principle, in such situations, we can calibrate every single measuring instruments – but that will cost a lot of money:

- many sensors are now very cheap, high school kids use a lot of sensors in their robotic projects, but
- calibrating each sensor means using a complex highly-accurate measuring instrument requiring lots of costly maintenance; it usually means bringing our sensor to a location where this standard instrument is placed; all this is very expensive.

Because of this high cost, most sensors are not individually calibrated, we do not know the corresponding probability distribution – we only know the upper bound on the absolute value of the measurement error, the upper bound provided by the manufacturer of this measuring instrument.

But why cannot we use probabilistic techniques? Situations when we do not know the exact probability distribution are common in statistics. In particular, there are many situations in which we have several alternatives, and we have no information about the probability of each alternative. Since we have no reason to assume that the probability of one alternative is larger than the probability of another alternative, a reasonable idea is to assign equal probabilities to all alternatives. This natural idea is known as *Laplace's Indeterminacy*

Principle, and it is a particular case of the more general *Maximum Entropy approach*; see, e.g., [4].

Similarly, if all we know is that a quantity is located somewhere on the interval, and we have no reason to conclude that some values from this interval are more probable and some are less probable, a reasonable idea is to assume that all the values from the interval are equally probable, i.e., that we have a uniform distribution on this interval. If we have several variables, and we have no information about their dependence, then it is reasonable to assume that they are independent – and this is exactly what the Maximum Entropy approach concludes in this case.

At first glance, this sounds like a reasonable approach, and in many problems, it works well, but, as we will show, in uncertainty quantification, this idea does not work. Indeed, let us consider a very simple example in which the algorithm f simply add up n values, i.e., when we compute $y = x_1 + \ldots + x_n$. Suppose that for each i, the result $\widetilde{x}_i$ of measuring the quantity x_i is simply 0, and the upper bound on the measurement error is $\Delta_i = 1$. This means that each value x_i is located somewhere on the interval $[-1, 1]$.

In this case, the sum y takes the largest value when each of the terms x_i attains its largest value 1, and this largest value is n. Similarly, the smallest possible value y is attained when each of the terms x_i attains its smallest value -1, so this smallest value of y is $-n$. Thus, the range of possible values of y is

$$[-n, n].$$

What if we use Laplace's principle? In this case, we assume that each x_i is uniformly distributed on the interval $[-1, 1]$, and all these variables are independent. Thus, for large n, according to the Central Limit Theorem (see, e.g., [14]), the distribution of the sum y is close to normal distribution. For each x_i the mean is 0 and the variance is $1/3$, so:

- the mean μ of the sum y is the sum of 0s, i.e., 0, and
- the variance of the sum y is the sum of the variances, i.e., $n/3$.

Thus, the standard deviation σ of y is proportional to $\sqrt{n}$.

It is known that with very high probability, all the values of a normally distributed random variable are located, in the interval $[\mu - k \cdot \sigma, \mu + k \cdot \sigma]$: e.g. for $k = 6$ the probability to be outside this interval is $\approx 10^{-8}$. Here, $\mu = 0$, $\sigma = c \cdot \sqrt{n}$ for some c, so we conclude that with high confidence, the value y is bounded by $6c \cdot \sqrt{n}$. But we know that y can take value n which, for large n, is much larger than const $\cdot \sqrt{n}$.

This shows that for the purposes of uncertainty quantification, we cannot just select one possible probability distribution our of many possible ones and ignore other possible distributions – this can drastically underestimate the inaccuracy of the results of data processing – and thus, potentially lead to a disaster.

Need to go beyond the probabilistic approach: imprecise probabilities. So far, we considered two types of possible situations:

- situations when we perform the full calibration and thus, we know the probability distribution of the measurement errors,
- situations when we do not perform any calibration, and thus, have no information about the corresponding probabilities – all we know is the interval of possible values of the measurement error.

In practice, we also sometimes encounter intermediate situations, when we perform *some* calibration, but not enough to uniquely determine the probability distributions. As a result, we only have partial information about the probabilities – i.e., we have several possible probability distributions which are all consistent with our calibration results. Such situations are known as situations with *imprecise probabilities*; see, e.g., [11] and references therein.

Need to go beyond the probabilistic approach: fuzzy case. In many practical situations, instead of measurement results, we get expert estimates. This is not typical in manufacturing, but this is a usual situation in medicine, in meteorology, in biology, etc. These estimates often use imprecise ("fuzzy") words from natural language like "small", "about 1.5", etc. It is desirable to use these expert estimates as inputs in data processing, but how? Computers are good in processing numbers, but, as everyone knows, they are not yet as good in processing natural language.

To handle such *fuzzy* knowledge, we need to use special techniques – which are known under the general name of *fuzzy techniques*; see, e.g., [5, 9, 12, 15].

What we do in this chapter. In this chapter, we briefly explain how (and why) to take these different types of beyond-traditional-probabilistic uncertainty into account when processing data.

2 Case of probabilistic uncertainty: reminder

Why do we need to describe the case of probabilistic uncertainty. To explain techniques for dealing with non-traditional types of uncertainty, let us first briefly recall how traditional probabilistic uncertainty is handled in data processing.

Data processing under probabilistic uncertainty: exact formulation of the problem. Probabilistic uncertainty means that we know:

- the data processing algorithm $y = f(x_1, \ldots, x_n)$;
- the measurement results $\widetilde{x}_1, \ldots, \widetilde{x}_n$; and
- the probability distributions $\rho_i(\Delta x_i)$ describing the probabilities of different values of the measurement error $\Delta x_i = \widetilde{x}_i - x_i$.

We want to find the probability distribution for the value $y = f(x_1, \ldots, x_n)$.

Bias is usually taken care of. In some cases, the mean value of the measurement error is different from 0. In mathematical terms, this means that the corresponding distribution has a *bias.* This happens: a clock may show time 2 minutes ahead, etc. In such situations, it makes sense to re-scale the readings of the measuring instrument – by subtracting this mean value. For example, for the clock, we subtract 2 minutes from all its readings.

As a result, we have a probability distribution in which the mean value of the measurement error is 0.

General case: Monte-Carlo simulations. For each i, we know the value $\widetilde{x}_i$, and we know the probability distribution $\rho_i(\widetilde{x}_i - x_i)$ for the difference between the measurement result $\widetilde{x}_i$ and the actual (unknown) value x_i of the corresponding quantity. Thus, the formula $\rho_i(\widetilde{x}_i - x_i)$ provides a probability distribution for x_i.

So, to find the probability distribution for y, we can do the following: several (N) times $k = 1, \ldots, N$:

- we simulate each random variable $\Delta x_i^{(k)}$ distributed according to the i-th given probability distribution $\rho_i(\Delta x_i)$;
- we simulate the values of x_i as $x_i^{(k)} = \widetilde{x}_i - \Delta x_i^{(k)}$; and
- we apply the data processing algorithm f to the simulated values, resulting in $y^{(k)} \stackrel{\text{def}}{=} f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)$.

Since the values $x_i^{(k)}$ follow the exact same probability distribution as the actual values x_i, the simulated values $y^{(k)}$ follow the exact same probability distribution as the desired quantity y. Thus, by analyzing the sample $y^{(1)}, \ldots, y^{(N)}$, we can find all the characteristics of the desired probability distribution for y.

This simulation-based technique, when we literally simulate the measurement errors, is known as *Monte-Carlo simulations.*

How accurate are Monte-Carlo simulations? It is known (see, e.g., [14]) that the relative accuracy ε of Monte-Carlo technique – as any accuracy of estimating a characteristic of a random variable from a sample of size N – is proportional to $\varepsilon \sim 1/\sqrt{N}$. Thus, to get the characteristics of $\Delta y = \widetilde{y} - y$ with accuracy ε, we need to perform $N \sim \varepsilon^{-2}$ simulations.

In particular, to get accuracy 10%, we need to perform $N = 100$ simulations.

Limitations of Monte-Carlo simulations. The data processing algorithm can be very complicated and time-consuming. To use Monta-Carlo simulations, we need to apply this algorithm N times. As a result, determining the accuracy of the result of data processing may require 100 time more computation time than computations themselves.

Even after such a long time, all we get is a very crude approximate estimation of accuracy – and to get a more accurate estimation, we need to perform even more computations! How can we perform computations faster?

Possibility of linearization. A possibility to speed up uncertainty estimations comes from the fact that measurement errors are usually relatively small:

$$\Delta x \ll x :$$

- rough measurements can have accuracy 10%,
- more accurate measurements can have accuracy 3%, 1%, and even higher.

The expressions $f(x_1, \ldots, x_n)$ are usually smooth. In such situations, we can expand the expression for

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

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

in Taylor series and keep only linear terms in this expansion – and ignore quadratic and higher order terms. Indeed:

- If $\Delta x_i \approx 10\%$, then $(\Delta x_i)^2 \approx 1\% \ll \Delta x_i$.
- If $\Delta x_i \approx 1\%$, then $(\Delta x_i)^2 \approx 0.01\% \ll \Delta x_i$, etc.

Thus, we get

$$\Delta y = \sum_{i=1}^{n} c_i \cdot \Delta x_i, \tag{1}$$

where we denoted $c_i \stackrel{\text{def}}{=} \dfrac{\partial f}{\partial x_i}$.

How this can speed up computations. In the formula (1), the only values depending on the data processing algorithm are partial derivatives c_i. In some cases, we have explicit formula for these partial derivatives. In general, we can estimate these partial derivatives by using numerical differentiation. Namely, by definition, the derivative is a limit

$$\frac{\partial f}{\partial x_i} = \lim_{h \to 0} \frac{f\left(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n\right) - \widetilde{y}}{h}.$$

Limit means that for small h, we have

$$c_i = \frac{\partial f}{\partial x_i} \approx \frac{f\left(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n\right) - \widetilde{y}}{h}, \tag{2}$$

and the smaller h, the higher the accuracy of this approximation. Estimation of c_i by using formula (2) is known as *numerical differentiation.*

To estimate all the value $c_1, \ldots, c_n$ this way, we need to call the algorithm f:

- first, to compute the estimate $\widetilde{y}$, and

- then n more times to compute the values

  $$f\left(\widetilde{x}_1, \ldots, \widetilde{x}_{i-1}, \widetilde{x}_i + h, \widetilde{x}_{i+1}, \ldots, \widetilde{x}_n\right)$$

  for $i = 1, \ldots, n$,

to the total of $n+1$ times.

When $n+1$ is smaller than the value N corresponding to desired accuracy – e.g., when $n+1 < 100$, the resulting computations are much faster than by using the general Monte-Carlo simulations.

Comment. Of course, in the linearized approach, we still need to run N (e.g., 100) simulations, but these simulations no longer require calling f, so they are fast.

Additional time saving: ubiquity of normal distributions. When n is large, and all error components are of the same order of magnitude, we can use the Central Limit Theorem, according to which the distribution of the sum of the large number of independent similarly distributed random variables is close to Gaussian (normal). (To be more precise, the theorem states that this distribution tends to normal when the number n of terms in this sum tends to infinity.)

Thus, for large n, we can safely conclude that the value Δy is normally distributed. Also, since all the components Δx_i have 0 mean, their linear combination Δy also has a zero mean. A normal distribution is uniquely determined by its mean and its standard deviation σ. Since here, the mean is 0, all we need to find is the standard deviation. This saves us computation time – since we only need to compute one charateristic of the probability distribution.

Another simplification comes from the fact that, based on the formula (1), we can conclude that

$$\sigma^2 = \sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2, \text{ hence } \sigma = \sqrt{\sum_{i=1}^{n} c_i^2 \cdot \sigma_i^2},$$

where σ_i is the standard deviation of Δx_i. Thus, there is no need for Monte-Carlo simulations here: once we compute σ_i, we can then use this explicit formula for σ – and, by the way, we will get the exact value of σ, while simulations will only lead to an approximate estimate.

Comment. The possibility to use normal distributions is not limited to the cases when n is large. In many cases, for each measuring instrument, the corresponding measurement error Δx_i itself comes from the joint effect of several error components of approximately the same size and is, thus, itself normal. In such situations, Δy is also normally distributed – as a linear combination of several independent normally distributed random variables.

3 Case of interval uncertainty

What is interval uncertainty: reminder. In some situations, all we know about each measurement error Δx_i is that it can take any value from the interval $[-\Delta_i, \Delta_i]$, and we do not have any information about the probability of different values from this interval. As a result, the actual value x_i of the corresponding quantity can take any value from the interval $[\widetilde{x}_i - \Delta_i, \widetilde{x}_i + \Delta_i]$.

Different values x_i from these intervals lead, in general, to different values of $y = f(x_1, \ldots, x_n)$. All we can compute is the range of possible values of y, i.e., the set

$$\{f(x_1, \ldots, x_n) : x_1 \in [\widetilde{x}_1 - \Delta_1, \widetilde{x}_1 + \Delta_1], \ldots, x_n \in [\widetilde{x}_n - \Delta_n, \widetilde{x}_n + \Delta_n]\}.$$

The data processing algorithm is usually continuous. The range of a continuous function on a connected set is also connected, so this range is also an interval $[\underline{y}, \overline{y}]$.

Computing the endpoints of this interval is known as *interval computations*; see, e.g., [3, 8, 10].

Let us formulate this problem in precise terms.

Data processing under interval uncertainty: exact formulation of the problem. Interval uncertainty means that we know:

- the data processing algorithm $y = f(x_1, \ldots, x_n)$;
- the measurement results $\widetilde{x}_1, \ldots, \widetilde{x}_n$; and
- the upper bounds Δ_i on the absolute values of the corresponding measurement errors $\Delta x_i = \widetilde{x}_i - x_i$.

We want to find the endpoints of the interval

$$[\underline{y}, \overline{y}] =$$
$$\{f(x_1, \ldots, x_n) : x_1 \in [\widetilde{x}_1 - \Delta_1, \widetilde{x}_1 + \Delta_1], \ldots, x_n \in [\widetilde{x}_n - \Delta_n, \widetilde{x}_n + \Delta_n]\}. \quad (3)$$

In general, this problem is not feasible. It is known that, in general, this problem is NP-hard – meaning that, unless P = NP (which most computer scientists believe to be impossible), no feasible algorithm can always provide a solution to this problem; see, e.g., [7].

In the linearized case, the problem becomes feasible. To solve the interval computations problem, we can take into account that, in practice, measurement errors are relatively small, and thus, for Δy, we can use a linearized formula (1). To find the range of possible values of Δy, we need to find the largest and the smallest values of this expression (1).

The sum attains its largest value when each of the terms $c_i \cdot \Delta x_i$ is the largest:

- When $c_i \geq 0$, the term $c_i \cdot \Delta x_i$ is an increasing function of Δx_i. Thus, its largest possible value is attained when $\Delta x_i \in [-\Delta_i, \Delta_i]$ attains its largest possible value $\Delta x_i = \Delta_i$. In this case, the term $c_i \cdot \Delta x_i$ takes the form
$$c_i \cdot \Delta_i.$$
- When $c_i \leq 0$, the term $c_i \cdot \Delta x_i$ is a decreasing function of Δx_i. Thus, its largest possible value is attained when $\Delta x_i \in [-\Delta_i, \Delta_i]$ attains its smallest possible value $\Delta x_i = -\Delta_i$. In this case, the term $c_i \cdot \Delta x_i$ takes the form
$$-c_i \cdot \Delta_i.$$

Both expressions can be described by a single formula $|c_i| \cdot \Delta_i$. Thus, the largest possible value of Δy is equal to

$$\Delta \overset{\text{def}}{=} \sum_{i=1}^{n} |c_i| \cdot \Delta_i. \tag{4}$$

Similarly, we can prove that the smallest possible value of Δy is equal to $-\Delta$. So, the range of Δy has the form $[-\Delta, \Delta]$. To find this range, it is sufficient to compute the value Δ.

A straightforward way to compute Δ. A straightforward way to compute Δ is to compute the partial derivatives by using the formula (2), and then use the formula (4).

Comment. This is similar to computing σ^2 in the case of normal distributions.

Limitations of the straightforward approach. When the number of inputs n is large, and the data processing algorithms is complicated, in the straightforward approach, to compute all partial derivatives, we need to call the algorithm f many $(n+1)$ times. For large n – e.g., when n is equal to several thousands – this is not realistic.

How can we compute Δ faster?

Cauchy-based techniques. We cannot directly use Monte-Carlo simulations – since we do not know the probability distributions for Δx_i. However, it turns out that we can use Monte-Carlo simulations *indirectly*, by using so-called *Cauchy distributions*, for which the probability density function is proportional to

$$\frac{1}{1 + \dfrac{x^2}{\Delta^2}}.$$

This distribution is presented in many statistics textbook, not as an example of something practically useful as many textbook examples of probability distributions, but usually as a pathological example – of a probability distribution for which standard deviation is infinite. However, in data processing under uncertainty, this distribution is very practically useful; see, e.g., [6, 11].

Specifically, what is useful is the following property of this distribution:

- if we have n independent random variables Δx_i which are Cauchy distributed with parameters Δ_i,
- then their linear combination (1) is also Cauchy distributed with the parameter described exactly by the formula (4)!

Thus, we can estimate Δ as follow: several (N) times $k = 1, \ldots, N$:

- we simulate each random variable $\Delta x_i^{(k)}$ distributed according to the Cauchy distribution with parameter Δ_i;
- we simulate the values of x_i as $x_i^{(k)} = \widetilde{x}_i - \Delta x_i^{(k)}$; and
- we apply the data processing algorithm f to the simulated values, resulting in $y^{(k)} \stackrel{\text{def}}{=} f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)$.

The resulting differences $\Delta y^{(k)} = y^{(k)} - \widetilde{y}$ are then Cauchy distributed with parameter Δ. To find this value, we can use Maximum Likelihood method – i.e., find the most probable value Δ. For Cauchy distribution, maximizing the corresponding probability is equivalent to solving the following easy-to-numerically-solve equation with one unknown Δ:

$$\sum_{k=1}^{N} \frac{1}{1 + \dfrac{\left(\Delta y^{(k)}\right)^2}{\Delta^2}} = \frac{N}{2}.$$

Comment. It is important to comment that, in contrast to Monte-Carlo simulations, when we simulate actual probability distributions, here, we are *not* simulating actual distributions. Indeed:

- we know that the measurement errors are located inside the intervals $[-\Delta_i, \Delta_i]$; but
- a Cauchy-distributed random variable can take, with positive probability, any real value – including values outside this interval.

In this case, the probability distributions are not real – they are a computational trick.

Which method should we use? We have described two methods for dealing with interval uncertainty:

- a straightforward method that needs n calls to the algorithm f – as many calls as there are inputs, and
- Cauchy method that needs $N \sim \varepsilon^{-2}$ calls to f, where ε is the relative accuracy with which we want to estimate Δ.

Clearly:

- if $n > N$, we should use the straightforward method, and
- if $n > N$, i.e., if we have many inputs, then we should use the Cauchy method.

4 Case of imprecise probabilities

Formulation of the problem. In the probabilistic case, we assume that we know the exact values of the parameters c_{ij} characterizing the distribution of measurement errors for each input x_i. Based on this information, we estimate the parameters $c_1, \ldots, c_m$ of the resulting y-distribution.

In some cases, we only have partial information about the probability distributions of each input x_i; e.g.:

- instead of the *exact* values of the parameters c_{ij} characterizing this distribution,
- we only know *bounds* on these values, i.e., we only know intervals $[\underline{c}_{ij}, \overline{c}_{ij}]$ that contain the actual values of these parameters.

Different values c_{ij} from the corresponding intervals lead, in general, to different values c_j. We thus need to find the ranges of possible values of these parameters c_j.

How can we find these ranges?

Natural idea. For each combination of values $c_{i1}, \ldots, c_{im}$ of the corresponding parameters, we can use one of the above probabilistic-case algorithms – e.g., Monte-Carlo approach or linearization – to find the values $c_1, \ldots, c_m$ of the parameters describing the resulting y-distribution. Thus, what we have, in effect, are new algorithms $F_1, \ldots, F_m$ that:

- given the values $x_1, \ldots, x_n$ of the inputs and the values c_{ij} of the parameters describing their uncertainty,
- estimate the parameter c_j describing the uncertainty of the result y of data processing:

$$c_j = F_j(x_1, \ldots, x_n, c_{11}, \ldots, c_{1m}, \ldots, c_{n1}, \ldots, c_{nm}).$$

In our case, we do not know the exact values of the parameters c_{ij}, we only know the intervals $[\underline{c}_{ij}, \overline{c}_{ij}]$ that contain these values. So, we can use one of the above-described interval-case technique to find, for each j, the range of the function F_j when $c_{ij} \in [\underline{c}_{ij}, \overline{c}_{ij}]$:

$$[\underline{c}_j, \overline{c}_j] = \{F(x_1, \ldots, x_n, \ldots, c_{ij}, \ldots) : c_{ij} \in [\underline{c}_{ij}, \overline{c}_{ij}]\}.$$

These are exactly the desired ranges for the parameters c_j.

5 Case of fuzzy uncertainty

Formulation of the problem: reminder. Sometimes, for the inputs x_i, instead of measurement results, we only have expert estimates, and these expert estimates are described not in terms of numbers, but in terms of imprecise ("fuzzy") words from a natural language.

Computers do not understand natural language well, so we need to translate this knowledge into numbers. How can we do it?

How to translate imprecise knowledge into numbers: a natural idea. For precise statement like "x_1 is positive", for each value x_1, this statement is either true or false. For example:

- this statement is true for $x_1 = 0.1$, and
- this statement is false for $x_1 = -0.1$.

For imprecise statement like "x_1 is small", for some values x_1, the expert him/herself is not 100% sure. A natural idea is to ask an expert, for all possible values x_1, to describe his/her degree of confidence in this statement on some scale – e.g., from 0 to 1, so that 0 means no confidence at all, while 1 means 100% confidence. This natural idea – first proposed by Lotfi Zadeh – is one of the main ideas of the techniques that he called *fuzzy logic*.

Of course, there are infinitely many possible values of each quantity x_i, so we cannot ask the expert infinitely many question, so what we *can* do is:

- ask about *some* values, and then
- perform some interpolation/extrapolation to cover values in between.

As a result, for each imprecise expert statement about x_i, we get a function $\mu_i(x_i)$ that assigns, to each possible value of the quantity x_i, the degree to which the expert is confident in this statement. This function is known as a *membership function*.

Need for logic. Suppose that we have two expert statements:

- a statement $S_1(x_1)$ about x_1 and
- a statement $S_2(x_2)$ about x_2.

We can extract, from the expert, the degrees of belief $\mu_1(x_1)$ and $\mu_2(x_2)$ in these statements corresponding to all possible values x_1 and x_2, but what we really need is the degree to which a combined statement "$S_1(x_1)$ and $S_2(x_2)$" is true.

Theoretically, we can ask an expert to evaluate the desired degree for all possible pairs (x_1, x_2), for all possible triples if we combine three statements, etc. However, in practice, this is not feasible. Even if we take only 10 possible values of each variable x_i, then, e.g., for five inputs we will need to ask $10^5 = 100000$ questions – this is not feasible.

Since we cannot directly ask for the expert's degree of confidence in complex statements like "A and B" (or in similar statements of the type "A or B"), we need to be able to estimate these degrees based on whatever information we have: namely, degrees of confidence a and b in statements A and B.

The algorithms computing the corresponding estimates $f_\&(a, b)$ for the degree of confidence in "A and B" and $f_\vee(a, b)$ for the degree of confidence in

"A or B" are known as "and"-operations and "or"-operations. For historical reasons, they are also known as *t-norm* and *t-conorm*.

The need for operations representing logical operations such as "and" and "or" is what caused Zadeh to call this approach fuzzy *logic*.

Which "and"- and "or"-operations should we choose? To select appropriate operations, let us list reasonable requirements. Let us start with the "and"-operation.

- First, the degree of confidence in a statement "A and B" – which is stronger than both A and B – cannot be larger than our degrees of confidence in each of the original statements A and B:
$$f_\&(a,b) \le a \text{ and } f_\&(a,b) \le b.$$
- Second, the statement "A and A" means the same as A, so we conclude that $f_\&(a,a) = a$.
- Third, if we increase degree of confidence on one or both statements A and B, this can only increase our degrees of confidence in the combined statement "A and B": if $a \le a'$ and $b \le b'$, then $f_\&(a,b) \le f_\&(a',b')$.

It turns out that these simple and natural requirements uniquely determine the choice of the "and"-operation. Indeed:

- Suppose that $a \le b$. Then, by the first property, $f_\&(a,b) \le a$, by the second property, $f_\&(a,a) = a$, and by the third property,
$$a = f_\&(a,a) \le f_\&(a,b).$$
From $f_\&(a,b) \le a$ and $a \le f_\&(a,b)$, we conclude that
$$f_\&(a,b) = a.$$
- Suppose that $b \le a$. Then, by the first property, $f_\&(a,b) \le b$, by the second property, $f_\&(b,b) = b$, and by the third property,
$$b = f_\&(b,b) \le f_\&(a,b).$$
From $f_\&(a,b) \le b$ and $b \le f_\&(a,b)$, we conclude that
$$f_\&(a,b) = b.$$

Both cases can be covered by a single expression $f_\&(a,b) = \min(a,b)$.

Let us now consider the "or"-operation. Here, we can also formulate three natural properties.

- First, the degree of confidence in a statement "A or B" – which is weaker than both A and B – cannot be smaller than our degrees of confidence in each of the original statements A and B:
$$a \le f_\vee(a,b) \text{ and } b \le f_\vee(a,b).$$

- Second, the statement "A or A" means the same as A, so we conclude that $f_\vee(a,a)=a$.
- Third, if we increase degree of confidence on one or both statements A and B, this can only increase our degrees of confidence in the combined statement "A or B": if $a\le a'$ and $b\le b'$, then $f_\vee(a,b)\le f_\vee(a',b')$.

It turns out that these simple and natural requirements uniquely determine the choice of the "or"-operation. Indeed:

- Suppose that $a\le b$. Then, by the first property, $b\le f_\vee(a,b)$, by the second property, $f_\vee(b,b)=b$, and by the third property,
$$f_\vee(a,b)\le f_\vee(b,b)=b.$$
From $b\le f_\vee(a,b)$ and $f_\vee(a,b)\le b$, we conclude that
$$f_\vee(a,b)=b.$$
- Suppose that $b\le a$. Then, by the first property, $a\le f_\vee(a,b)$, by the second property, $f_\vee(a,a)=a$, and by the third property,
$$f_\vee(a,b)\le f_\vee(a,a)=a.$$
From $a\le f_\vee(a,b)$ and $f_\vee(a,b)\le a$, we conclude that
$$f_\vee(a,b)=a.$$

Both cases can be covered by a single expression $f_\vee(a,b)=\max(a,b)$.

Thus, we select $f_\&(a,b)=\min(a,b)$ and $f_\vee(a,b)=\max(a,b)$.

Towards data processing under fuzzy uncertainty. Suppose that we know the algorithm $y=f(x_1,\ldots,x_n)$ that is used in data processing, and our information about each input x_i is described by a membership function $\mu_i(x_i)$. We want to describe the degree $\mu(y)$ to which different values of y are possible.

A value y is possible if there exist values $x_1,\ldots,x_n$ for which $y=f(x_1,\ldots,x_n)$ and for which "x_1 is possible *and* x_2 is possible, etc.", i.e., if:

- either the statement in quotes holds for one tuple $(x_1,\ldots,x_n)$ for which $y=f(x_1,\ldots,x_n)$,
- *or* this statement holds for another tuple $(x_1,\ldots,x_n)$ for which $y=f(x_1,\ldots,x_n)$, etc.

For each i, the degree to which each value x_i is possible is equal to $\mu_i(x_i)$. Here, "and" is represented by minimum, so the degree to which the statement in quotes in satisfied is equal to $\min(\mu_1(x_1),\ldots,\mu_n(x_n))$.

Here, "or" is represented by maximum, so the desired degree is equal to
$$\mu(y)=\max\{\min(\mu_1(x_1),\ldots,\mu_n(x_n)) : f(x_1,\ldots,x_n)=y\}. \tag{5}$$

This formula – first proposed by Zadeh – is known as *Zadeh's extension principle.*

How can we use this principle for computations? The idea comes from the fact that while expert's conclusions are often imprecise, we need to eventually make decisions. We can rarely achieve full confidence, so a natural idea is to select some threshold α and to make a decision if the degree of confidence is greater than or equal to α. The set of all such values $\mathbf{x}(\alpha) \stackrel{\text{def}}{=} \{x : \mu(x) \geq \alpha\}$ is known as the *α-cut* of the corresponding membership function.

Once we know all α-cuts, we can then uniquely determine the original membership function $\mu(x)$ – namely, for each x, the value $\mu(x)$ is the smallest value α for which $x \in \mathbf{x}(\alpha)$. Thus, instead of describing the expert's statement by a membership function, we can describe it by listing α-cuts corresponding to different levels α.

Usually, an expert can only provide a degree from 0 to 1 with accuracy 0.1 – e.g., it is difficult to distinguish between degrees of confidence 0.30 and 0.31. Thus, it is sufficient to only describe α-cuts corresponding to

$$\alpha = 0, \quad \alpha = 0.1, \quad \alpha = 0.2, \ldots, \alpha = 0.9, \quad \alpha = 1.0.$$

For most natural language words, the degree of confidence first increases then decreases. For such membership functions, each α-cut is an interval.

It turns out that α-cuts are useful in processing fuzzy inputs. Indeed, when is $\mu(y) \geq \alpha$, i.e., when is $y \in (\alpha)$? According to the formula (5), this means that there exists some values $x_1, \ldots, x_n)$ for which $y = f(x_1, \ldots, x_n)$ and for which $\min(\mu_1(x_1), \ldots, \mu_n(x_n)) \geq \alpha.$, The last inequality, in its turn, means that $\mu_i(x_i) \geq \alpha$ for all i, i.e., that $x_i \in \mathbf{x}_i(\alpha)$. Thus, $y \in \mathbf{y}(\alpha)$ means that $y = f(x_1, \ldots, x_n)$ for some $x_i \in \mathbf{x}_i(\alpha)$. In other words, the α-cut for y is the range of the function $f(x_1, \ldots, x_n)$ on the α-cuts for x_i:

$$\mathbf{y}(\alpha) = \{f(x_1, \ldots, x_n) : x_1 \in \mathbf{x}_1(\alpha), \ldots, x_n \in \mathbf{x}_n(\alpha)\}. \tag{6}$$

We already know how to compute this range, so we arrive at the following algorithm for processing fuzzy uncertainty.

How to process fuzzy uncertainty: algorithm. We know the data processing algorithm $f(x_1, \ldots, x_n)$, and we know the membership functions $\mu_i(x_i)$ describing the expert's knowledge of the inputs.

Then, for each $\alpha = 0, 0.1, \ldots, 0.9, 1.0$:

- First, for each i, we compute the corresponding α-cuts. These α-cuts will be intervals.
- Then, we use one of the above-described interval methods to compute the range (6).

The resulting α-cuts $\mathbf{y}(\alpha)$ describe the resulting information about the desired quantity y.

6 How to take uncertainty into account in machine learning – especially in deep learning

Formulation of the problem. The above techniques assume that we know the equations that describe the system's behavior and the system's dynamics – and thus, we have an algorithm $f(x_1, \ldots, x_n)$ that estmates the desired quantity y based on the known values $x_1, \ldots, x_n$.

In many practical situations, however, we do not have this knowledge, we must determine the system's dynamics based on its observed behavior. This is the subject of *machine learning*; see, e.g., [1, 2]. In this case:

- in several cases $k = 1, \ldots, K$, we know the values $x_1^{(k)}, \ldots, x_n^{(k)}$, and $y^{(k)}$;
- based on these values, the machine learning algorithm find an algorithm $f(x_1, \ldots, x_n)$ for which

  $$y^{(k)} \approx f\left(x_1^{(k)}, \ldots, x_n^{(k)}\right)$$

  for all k.

At present, one of the main tools for machine learning is deep learning; see, e.g., [2] and references therein.

A natural question is: how do we take into account uncertainty with which we know the inputs?

Straightforward approach and its limitations. Once we have an algorithm $f(x_1, \ldots, x_n)$ – whether it is explicitly given or presented as a neural network – we can apply one of the above techniques to take care of the input's uncertainty.

The limitation of this straightforward approach is that it requires to run the neural network several times – so the time needed to take uncertainty into account is much longer than the time needed to compute the estimate itself.

How can we speed up this process?

Natural idea. Once we have trained the neural network – or any other appropriate tool – to estimate y, i.e., once we come up wth an appropriate algorithm $f(x_1, \ldots, x_n)$, we can then use one of the above algorithms, on several cases, to estimate how uncertainty in the inputs lead to uncertainty in the result. In all these cases, for different values $x_i^{(k)}$ of the input and for different parameters $c_1^{(k)}, \ldots, c_n^{(k)}$ of the corresponding uncertainty (standard deviations, upper bounds, etc.), we get not only the estimate $y^{(k)}$ for the quantity y, we also get an estimate $c^{(k)}$ for the resulting uncertainty in y.

Then, a natural idea is to train a new neural network $c = F(x_1, \ldots, x_n, c_1, \ldots, c_n)$ so that:

- given the inputs $x_1, \ldots, x_n$ and the information $c_1, \ldots, c_n$ about the uncertainty of these inputs,
- this neural network will return the corresponding parameters c.

In other words, we want to find a function F for which

$$c^{(k)} = F\left(x_1^{(k)}, \ldots, x_n^{(k)}, c_1^{(k)}, \ldots, c_n^{(k)}\right)$$

for all k.

Then, to find the corresponding uncertainty, you will need to run a neural network – namely, the neural network corresponding to F – only once.

Comment. Most of the above techniques involve applying the algorithm f not only to the original results $x_i^{(k)}$ but also to perturbed values. This may help not only to estimate uncertainty: it will also help to avoid a big problem of deep learning, that often, a minor change in the input drastically changes the result. For example, changing a small number of pixels in the image of a cat can lead the network to conclude that it is a dog; see, e.g., [2].

Good news is that such confusing perturbations are rare, and often, this problem disappears if we add additional noise. So, if we train the neural network not only on the original values x_i, but also on perturbed values, this may not only help to estimate the resulting uncertainty, it will also help to avoid wrong results.

Acknowledgments

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).

The authors is greatly thankful to the editors for their encouragement.

References

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

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Leaning*, MIT Press, Cambridge, Massachusetts, 2016.

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

[4] E. T. Jaynes and G. L. Bretthorst, *Probability Theory: The Logic of Science*, Cambridge University Press, Cambridge, UK, 2003.

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

[6] 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.

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

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

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

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

[11] H. T. Nguyen, V. Kreinovich, B. Wu, and G. Xiang, *Computing Statistics under Interval and Fuzzy Uncertainty*, Springer Verlag, Berlin, Heidelberg, 2012.

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

[13] S. G. Rabinovich, *Measurement Errors and Uncertainties: Theory and Practice*, Springer, New York, 2005.

[14] D. J. Sheskin, *Handbook of Parametric and Non-Parametric Statistical Procedures*, Chapman & Hall/CRC, London, UK, 2011.

[15] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.