How Interval Measurement Uncertainty Affects the Results of Data Processing: A Calculus-Based Approach to Computing the Range of a Box

Andrew Pownuk and Vladik Kreinovich
Computational Science Program
University of Texas at El Paso, El Paso, TX 79968, USA
ampownuk@utep.edu, vladik@utep.edu

Home Page
Title Page
Page 1 of 25
Go Back
Full Screen
Close
Quit

1. Need for Indirect Measurements

- In many practical situations:
  - we are interested in the values of the quantities $y_1, \ldots, y_m$
  - which are difficult – or even impossible – to measure directly.
- Since we cannot measure these quantities directly, a natural idea is to measure them *indirectly*, i.e.:
  - to measure related quantities $x_1, \ldots, x_n$ which are related to y_j by known relations, an
  - to use appropriate algorithms to find the values of the desired quantities:

$$y_1 = f_1(x_1, \ldots, x_n); \ldots, y_m = f_m(x_1, \ldots, x_n).$$

- *Comment.* In the real world, the relations are usually smooth.

2. Need to Take into Account Measurement Uncertainty

- In practice, measurements are never absolutely precise.
- The measurement result $\widetilde{x}_i$ is, in general, different from the actual (unknown) values of the corr. quantity.
- When we plug in $\widetilde{x}_i \neq x_i$, we, in general, get the values $\widetilde{y}_j = f_j(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ which are different from y_j.
- How can we gauge the resulting uncertainty in y_j?

3. Case of Interval Measurement Uncertainty

- In many practical situations:
  - the only information that we have about the measurement error $\Delta x_i \stackrel{\text{def}}{=} \widetilde{x}_i - x_i$
  - is the upper bound Δ_i provided by the manufacturer of the corresponding measuring instrument.
- If the manufacturer provide no such bound, then
  - it is not a measuring instrument,
  - it is a device for producing wild guesses.
- In this case:
  - once we know the measurement result $\widetilde{x}_i$,
  - the only information we have about x_i is that it is somewhere on the interval $[\underline{x}_i, \overline{x}_i]$, where

$$\underline{x}_i \stackrel{\text{def}}{=} \widetilde{x}_i - \Delta_i \text{ and } \overline{x}_i \stackrel{\text{def}}{=} \widetilde{x}_i + \Delta_i.$$

4. Case of Interval Uncertainty (cont-d)

- There is no a priori known relation between x_i's.
- So, the set of all possible values of x_i should not depend on the values of all other quantities x_j, $j \neq i$.
- Thus, the set of all possible values of the tuple $x = (x_1, \ldots, x_n)$ is the box $[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$.

5. Resulting Problem

- Once we know that x belongs to the box, what are the possible values of the tuple $y = (y_1, \ldots, y_m)$?
- In mathematical terms, what is the range of the box under the mapping f?
- In this talk, we describe calculus-based techniques for solving this problem.

6. Simplest Case When We Have Only One Desired Quantity y_1: Analysis of the Problem

- Let us start with the simplest case, when we have only one desired quantity y_1.
- In this case, we are interested in the range of the function $f_1(x_1, \ldots, x_n)$ when each x_i is in $[\underline{x}_i, \overline{x}_i]$.
- For smooth (even for continuous) functions, this range is connected and is, thus, an interval $[\underline{y}_1, \overline{y}_1]$, where:
  - $\underline{y}_1$ is the smallest possible value of the function $f_1(x_1, \ldots, x_n)$ on the given box, and
  - $\overline{y}_1$ is the largest possible value of $f_1(x_1, \ldots, x_n)$ on the given box.

7. Simplest Case When We Have Only One Desired Quantity y_1 (cont-d)

- For each variable x_i, the maximum (or minimum) of the expression $y_1 = f_1(x, \ldots, x_n)$ is attained:
  - either at one of the endpoints of this interval, i.e., for $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$,
  - or inside the corresponding interval $(\underline{x}_i, \overline{x}_i)$.
- According to calculus:
  - if the maximum or minimum is attained inside an interval,
  - then the corresponding derivative $\dfrac{\partial f_1}{\partial x_i}$ is $= 0$.

8. Simplest Case When We Have Only One Desired Quantity y_1 (cont-d)

- So, for each i, it is sufficient to consider three possible cases:
  - the case when $x_i = \underline{x}_i$;
  - the case when $x_i = \overline{x}_i$, and
  - the case when $\dfrac{\partial f_1}{\partial x_i} = 0$.
- Thus:
  - to find the minimum $\underline{y}_1$ and the maximum $\overline{y}_1$ of the function $y_1 = f_1(x_1, \ldots, x_b)$ over the box,
  - it is sufficient to consider all possible combinations of these 3 cases.
- In other words, we arrive at the following algorithm.

9. Case When We Have Only One Desired Quantity y_1: Algorithm

- Consider all systems of equations, in which, for each i, we have one of the three alternatives:

$$x_i = \underline{x}_i, \quad x_i = \overline{x}_i, \text{ and } \frac{\partial f_1}{\partial x_i} = 0.$$

- There are 3^n such systems.
- For each of these systems:
  - we find the corresponding values $x = (x_1, \ldots, x_n)$ and
  - we compute the corresponding value $y_1 = f(x_1, \ldots, x_n)$.
- The largest of thus computed values is $\overline{y}_1$, the smallest is $\underline{y}_1$.

10. Discussion

- This algorithm requires solving an exponential number of systems and thus takes exponential time.
- This is, however, unavoidable, since it is known that:
  - already for quadratic functions $f_1(x_1, \ldots, x_n)$,
  - the problem of computing the bounds $\underline{y}$ and $\overline{y}$ is NP-hard.
- This means that:
  - unless P=NP (which most computer scientists believe to be impossible),
  - super-polynomial (e.g., exponential) computation time is unavoidable – at least for some inputs.

11. Discussion (cont-d)

- Exponential time does not mean that the algorithm is not practical.
- For reasonably small n, solving 3^n system is quite reasonable.
- For example, for $n = 10$, we need to solve less than 60,000 systems.
- It is a large number, but it is quite doable.
- For $n = 15$, we need to solve about 5 million systems – still possible.

12. What We Plan to Do Next

- In the following, we show how we can extend this calculus-based approach to the general case.
- We thus reduce:
  - the difficult-to-solve problem of finding the range
  - to more well-studied problems of solving systems of equations.

13. Case When the Number m of Desired Quantities Is = the Number n of Auxiliary Ones

- To find the range means to find its border.
- At almost all points on the border, there is – locally – at least one tangent plane.
- A plane in an m-dimensional space has the form

$$\sum_{j=1}^{m} c_j \cdot y_i = c_0.$$

- Thus, at this border point $y = (y_1, \ldots, y_m)$, the following linear expression attains its local max or min:

$$y = \sum_{j=1}^{m} c_j \cdot y_j = f(x_1, \ldots, x_n) \stackrel{\text{def}}{=} \sum_{j=1}^{m} c_j \cdot f_j(x_1, \ldots, x_n)$$

14. Case When $m = n$ (cont-d)

- Similarly to the previous case, this may mean that one of the inputs x_i:
  - either attains its largest possible value $\overline{x}_i$
  - or its smallest possible value $x_i = \underline{x}_i$.
- In this case, the corresponding condition $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$ determines the $(n-1)$-dimensional set.
- This set could be part of the border.
- It may also mean that the max or min of the linear function is attained when all x_i are inside.
- In this case, we get $\dfrac{\partial f}{\partial x_i} = 0$ for all i, i.e., we get

$$\sum_{j=1}^{m} c_j \cdot \frac{\partial f_j}{\partial x_i} = 0 \text{ for all } i.$$

15. Case When $m = n$ (cont-d)

- We get $\sum_{j=1}^{m} c_j \cdot \dfrac{\partial f_j}{\partial x_i} = 0$ for all i.
- In algebraic terns, the existence of $c_j \neq 0$ means that $m = n$ gradient vectors are linearly dependent:

$$\left(\frac{\partial f_j}{\partial x_1}, \ldots, \frac{\partial f_j}{\partial x_n}\right).$$

- According to linear algebra, this means that the determinant of the Jacobian matrix is equal to 0:

$$\det \left\| \frac{\partial f_j}{\partial x_i} \right\| = 0.$$

- So, we arrive at the following algorithm.

16. Case When $m = n$: Algorithm

- To find the border of the desired range, for each i from 1 to $m = n$, we form two systems of equations:
  - the system $y_j = f_j(x_1, \ldots, x_n)$ in which we substitute $x_i = \underline{x}_i$, and
  - the system in which we substitute $x_i = \overline{x}_i$.
- Each of these systems provides a set of co-dimension 1 that could potentially serve as part of the border.
- To these possible border sets, we add the set corresponding to the equation $\det(\partial f_j/\partial x_i) = 0$.
- This equation defined a set of co-dimension 1.
- Plugging this set into $y_j = f_j(x_1, \ldots, x_n)$, we get a y-set of co-dimension one.
- This set can also be part of the border.

17. Case When $m = n$: Algorithm (cont-d)

- We know that the actual border can contain only segments of the above type.
- So once we have computed all these segments, we can reconstruct the border.

18. General Case: Analysis of the Problem

- We have already considered the case when $m = n$.
- There are two remaining cases: when $n < m$ and when $m < n$.
- When $n < m$, the set of all possible values of the tuple y is of of smaller dimension than the m.
- So, this set is its own boundary.
- Let us now consider the case when $m < n$.
- In this case, also, some linear combination attains its max or min:

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{m} c_j \cdot f_j(x_1, \ldots, x_n).$$

- Let v denote the number of inputs x_i for which at this max-or-min point, we have $x_i = \underline{x}_i$ or $x_i = \overline{x}_i$.

19. General Case (cont-d)

- For each of the remaining $n-v$ variables x_i, we then have the equation
$$\sum_{j=1}^{m} c_j \cdot \frac{\partial f_j}{\partial x_i} = 0.$$
- This equality must hold for all $(n-v)$ values of i, so we must have $(n-v)$ equations.
- We can select one of the values c_j equal to 1. Then:
  - the other $m-1$ values of c_j can be determining
  - if we consider the first $m-1$ conditions as a system of linear equations with $m-1$ unknowns.
- We substitute these values for c_j into the remaining $n-v-(m-1)$ equalities.
- We thus get $n-v-(m-1)$ equalities that relate $n-v$ unknowns.

20. General Case (cont-d)

- In general, each additional equality imposed on elements of a set decreases its dimension by 1.
- For example, in the 3-D space:
  - the set of all the points that satisfy a certain equality is usually a 2-D surface,
  - the set of points that satisfy two independent equalities in a 1-D line, etc.
- In our case:
  - the dimension of the set of all the $(n-v)$-dimensional tuples x that satisfy all $n-v-(m-1)$ equalities
  - is equal to the difference

$$(n-v)-(n-v-(m-1)) = m-1.$$

- The image of this $(m-1)$-dimensional set under the transformation $y_j = f_j(x)$ is also $(m-1)$-dimensional.

21. General Case (cont-d)

- The image of a $(m-1)$-dimensional set under the transformation $y_j = f_j(x)$ is also $(m-1)$-dimensional.
- So it forms a surface in the m-dimensional space of all possible tuples $y = (y_1, \ldots, y_m)$.
- As a result, we get the following algorithm.

22. General Case: Algorithm

- We consider all possible subsets I of the set $\{1,\ldots,n\}$ of all indices of the inputs x_i.
- For each such subset I of size v, we consider all 2^v possible combinations of values $\underline{x}_i$ and $\overline{x}_i$.
- For each such combination, we consider the following system of equations for all $i\notin I$:

$$\sum_{j=1}^{m} c_j\cdot\frac{\partial f_j}{\partial x_i}=0.$$

- We can set up one of the values c_j to 1 and the first $m-1$ equations to describe c_j as a function of $x_1,\ldots,x_m$.
- We substituting the resulting expressions for c_j in terms of x_i into the remaining $n-v-(m-1)$ equalities.
- We thus get a $(m-1)$-dimensional set of tuples x.

23. General Case: Algorithm (cont-d)

- Substituting this set of tuples into the formula $y_j = f_j(x)$, we get a $(m-1)$-dimensional set of y-tuples.
- We thus get several $(m-1)$-dimensional sets.
- We know that the actual border can only consist of the above fragments.

24. Acknowledgments

This work was supported in part by the National Science Foundation grant HRD-1242122 (Cyber-ShARE Center).