Computing the Range of a Function-of-Few-Lin.-Combinations Under Linear Constraints: A Feasible Algorithm

Salvador Robles, Martine Ceberio, and Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso
El Paso, Texas 79968, USA
sroblesher1@miners.utep.edu, mceberio@utep.edu, vladik@utep.edu

1. First case study: Fourier transform

- In many application areas, an important data processing technique is Fourier transform.
- It transforms, e.g., the values $x_0, x_1, \ldots, x_{n-1}$ of a certain quantity at several moments of time into *Fourier coefficients*

$$X_k = \sum_{i=0}^{n-1} x_i \cdot \exp\left(-\mathrm{i} \cdot \frac{2\pi \cdot k \cdot i}{n}\right), \text{ where } \mathrm{i} \stackrel{\text{def}}{=} \sqrt{-1}.$$

- Since $\exp(i \cdot x) = \cos(x) + \mathrm{i} \cdot \sin(x)$, the value X_k can be written as $X_k = A_k + \mathrm{i} \cdot B_k$, where

$$A_k = \sum_{i=0}^{n-1} x_i \cdot \cos\left(\mathrm{i} \cdot \frac{2\pi \cdot k \cdot i}{n}\right) \text{ and } B_k = -\sum_{i=0}^{n-1} x_i \cdot \sin\left(\mathrm{i} \cdot \frac{2\pi \cdot k \cdot i}{n}\right).$$

- It is also important to know the absolute value (modulus) $M_k \stackrel{\text{def}}{=} |X_k| = \sqrt{A_k^2 + B_k^2}$.

2. Need for interval uncertainty

- The values x_i come from measurements, and measurements are never absolutely exact.
- The measurement result $\widetilde{x}_i$ is, in general, different from the actual (unknown) value x_i of the corresponding quantity.
- Often, 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 on $|\Delta x_i|$.
- In such situations, after the measurement, the only information that we have about the actual value x_i is that

$$x_i \in [\underline{x}_i, \overline{x}_i], \text{ where } \underline{x}_i = \widetilde{x}_i - \Delta_i \text{ and } \overline{x}_i = \widetilde{x}_i + \Delta_i.$$

3. Need to estimate the ranges under interval uncertainty

- In general, processing data $x_0, \ldots, x_{n-1}$ means applying an appropriate algorithm $f(x_0, \ldots, x_{n-1})$ to compute the desired value

$$y = f(x_0, \ldots, x_{n-1}).$$

- In the case of interval uncertainty:
  - for different possible values $x_i \in [\underline{x}_i, \overline{x}_i]$
  - we have, in general, different possible values of

$$y = f(x_0, \ldots, x_{n-1}).$$

  It is therefore desirable to find the range of possible values of y:

$$[\underline{y}, \overline{y}] \stackrel{\text{def}}{=} \{f(x_0, \ldots, x_{n-1}) : x_i \in [\underline{x}_i, \overline{x}_i] \text{ for all } i\}.$$

- In particular, it is desirable to compute such ranges for the values A_k, B_k, and M_k corresponding to Fourier transform.

4. Ranges of Fourier coefficients: what is known

- The values A_k and B_k are linear functions of the quantities x_i.
- For a linear function $y = c_0 + \sum\limits_{i=0}^{n-1} c_i \cdot x_i$, the range is easy to compute: it is equal to

$$[\widetilde{y} - \Delta, \widetilde{y} + \Delta], \text{ where } \widetilde{y} = c_0 + \sum_{i=0}^{n-1} c_i \cdot \widetilde{x}_i \text{ and } \Delta = \sum_{i=0}^{n-1} |c_i| \cdot x_i.$$

- The problem of computing the range of M_k – or, what is equivalent, the range of its square M_k^2 – is more complicated.
- Indeed, M_k^2 is a quadratic function of the inputs.
- In general, for quadratic functions, the problem of computing the range under interval uncertainty is NP-hard.
- However, for the specific case of M_k^2, feasible algorithms are known.
- Reminder: an algorithm is called feasible if its computation time is bounded by a polynomial of n.

5. Need to take constraints into account

- We know the ranges $[\underline{x}_i, \overline{x}_i]$ for each quantity x_i.
- We also often know that the actual values x_i do not change too fast.
- E.g., we may know that the consequent values x_i and x_{i+1} cannot change by more than some small value ε: $-\varepsilon \le x_{i+1} - x_i \le \varepsilon$.

6. Motivation: second case study

- In the previous example, we had a nonlinear function – namely, the sum of two squares – applied to linear combinations of the inputs.
- There is another important case when a nonlinear function is applied to such a linear combination.
- Namely, data processing in an artificial neural network:
  - a nonlinear function $s(z)$ – known as *activation function* –
  - is applied to a linear combination $z = \sum\limits_{i=0}^{n-1} w_i \cdot x_i + w_0$ of the inputs $x_0, \ldots, x_{n-1}$,
  - resulting in the output signal $y = s\left(\sum\limits_{i=0}^{n-1} w_i \cdot x_i + w_0\right)$.

7. General formulation of the problem

- In both cases studies, we have a function $f(x_0, \ldots, x_{n-1})$ which has the form $f(x_0, \ldots, x_{n-1}) = F(y_1, \ldots, y_k)$.
- Here, k is much smaller than n (this is usually denoted by $k \ll n$), and each y_j is a linear combination of the inputs

$$y_j = \sum_{i=0}^{n-1} w_{j,i} \cdot x_i + w_{j,0}.$$

- In the Fourier coefficient case, $k = 2$, and $F(y_1, y_2) = \sqrt{y_1^2 + y_2^2}$.
- In the case of a neuron, $k = 1$, and $F(y_1)$ is the activation function. We know the intervals $[\underline{x}_i, \overline{x}_i]$ of possible values of all the inputs x_i.
- We may also know some linear constraints of the input values, i.e., constraints of the form

$$\sum_{i=0}^{n-1} c_{a,i} \cdot x_i \le c_{a,0}, \quad c_{b,0} \le \sum_{i=0}^{n-1} c_{b,i} \cdot x_i, \text{ or } \sum_{i=0}^{n-1} c_{d,i} \cdot x_i = c_{d,0}.$$

8. General formulation of the problem (cont-d)

- We want to find the range $[\underline{y}, \overline{y}]$ of the function f under all these constraints:
  - interval constraints $x_i \in [\underline{x}_i, \overline{x}_i]$ and
  - additional linear constraints.
- So, we want to find the following interval

$$[\underline{y}, \overline{y}] = \left\{ F\left(\sum_{i=0}^{n-1} w_{0,i} \cdot x_i + w_{0,0}, \ldots, \sum_{i=0}^{n-1} w_{n-1,i} \cdot x_i + w_{n-1,0} \right) : x_i \in [\underline{x}_i, \overline{x}_i], \right.$$

$$\left. \sum_{i=0}^{n-1} c_{a,i} \cdot x_i \le c_{a,0}, \quad c_{b0} \le \sum_{i=0}^{n-1} c_{bi} \cdot x_i, \quad \sum_{i=0}^{n-1} c_{d,i} \cdot x_i = c_{d,0} \right\}.$$

- We design a feasible algorithm that computes the desired range – under some reasonable conditions on $F(y_1, \ldots, y_k)$.

9. What are reasonable conditions on the function $F(y_1, \ldots, y_k)$: discussion

- We want to come up with a feasible algorithm for computing the desired range of the function $f(x_0, \ldots, x_{n-1})$.
- In general, in computer science, feasible means:
  - that the computation time t should not exceed a polynomial of the size of the input,
  - i.e., equivalently, that $t \le v \cdot n^p$ for some values v and p.
- Otherwise:
  - if this time grows faster, e.g., exponentially,
  - then, for reasonable values n, we will require computation times longer than the lifetime of the Universe.
- For computations with real numbers, it is also reasonable to require that the value of the function does not grow too fast.

10. What are reasonable conditions on the function $F(y_1, \ldots, y_k)$: discussion (cont-d)

- In other words, this value should be bounded by a polynomial of the values of the inputs.
- It is also necessary to take into account:
  - that, in general, the value of a real-valued function can be only computed with some accuracy $\varepsilon > 0$, and
  - that the inputs x_i can also only be determined with some accuracy $\delta > 0$.
- Thus, it is also reasonable to require that:
  - the time needed to compute the function with accuracy ε
  - is bounded by some polynomial of ε (and of the inputs).

11. What are reasonable conditions on the function $F(y_1, \ldots, y_k)$: discussion (cont-d)

- Also:
  - the accuracy δ with which we need to know the inputs to compute the value of the function with desired accuracy ε
  - should also be bounded from below by some polynomial of ε (and of the inputs).
- We want to make sure that the function $f(x_0, \ldots, x_{n-1})$ has these "regularity" properties.
- So, we need to restrict ourselves to functions $F(y_1, \ldots, y_k)$ that have similar regularity properties.
- Otherwise:
  - if even computing a single value $F(y_1, \ldots, y_k)$ is not feasible,
  - we cannot expect computation of the range of this function to be feasible either.

12. Resulting definition

- Let $T \stackrel{\rm def}{=} (v_F, p_F, v_a, p_a, q_a, t_c, p_c, q_c)$ be a tuple of real numbers.
- We say that a function $F(y_1, \ldots, y_k)$ is T-*regular* if the following conditions are satisfied:
  - for all inputs y_j, we have $|F(y_1, \ldots, y_k)| \le v_F \cdot (\max |y_j|)^{p_v}$;
  - for each $\varepsilon > 0$, if $|y_j - y'_j| \le \delta \stackrel{\rm def}{=} v_a \cdot (\max |y_j|)^{p_a} \cdot \varepsilon^{q_a}$, then

    $$|F(y_1, \ldots, y_k) - F(y'_1, \ldots, y'_k)| \le \varepsilon;$$

  - there exists an algorithm that, give inputs y_j and $\varepsilon > 0$, computes the value $F(y_1, \ldots, y_k)$ with accuracy $\varepsilon > 0$ in time

    $$T_f(y_1, \ldots, y_k) \le t_c \cdot (\max |y_j|)^{p_c} \cdot \varepsilon^{q_c}.$$

- *Comment:* One can easily check that the Fourier-related function $F(y_1, y_2) = \sqrt{y_1^2 + y_2^2}$ is T-regular for an appropriate tuple T.

13. Main problem

- Let T be a tuple, let $F(y_1, \ldots, y_k)$ be a T-regular function, and let W, X and ε be real numbers.
- By a *problem of computing the range of a function-of-few-linear-combinations under linear constraints*, we mean the following:
- *Given:*
  - a function $F(y_1, \ldots)$, where $y_i = \sum\limits_j w_{ij}$ and $|w_{j,i}| \le W$ for all i, j,
  - n intervals $[\underline{x}_i, \overline{x}_i]$ for which $|\underline{x}_i| \le X$ and $|\overline{x}_i| \le X$ for all i, and
  - m linear constraints.
- *Compute:*
  - the ε-approximation to the range $[\underline{y}, \overline{y}]$ of this function
  - for all tuples of values x_i from the given intervals that satisfy the given constraints.

14. Main result

- **Proposition.**
  - *For each tuple T, for each T-regular function, and for each selections of values W, X, and ε,*
  - *there exists a feasible algorithm for computing the range of a function-of-few-linear-combinations under linear constraints.*
- In other words, we have an algorithm that finishes computations in time bounded by a polynomial of n and m.

15. Proof

- We have $|\underline{x}_i| \leq X$ and $|\overline{x}_i| \leq X$.
- So, we conclude that for all values $x_i \in [\underline{x}_i, \overline{x}_i]$, we have $|x_i| \leq X$.
- Since $|w_{j,i}| \leq W$, we conclude that $|y_j| \leq n \cdot W \cdot X + W$, hence $\max |y_j| \leq n \cdot W \cdot X + W$.
- To compute the value of the function $F(y_1, \ldots, y_k)$ with the desired accuracy ε, we need know each y_k with accuracy $\delta \sim \max |y_j|)^{p_a}$.
- In view of the above estimate for $\max |y_j|$, we need $\delta \sim n^a$.
- We can divide each interval $[-(n \cdot W \cdot X + W), n \cdot W \cdot X + W]$ of possible values of y_j into sub-intervals of size 2δ.
- There will be $\dfrac{2(n \cdot W \cdot X + W)}{2\delta} \sim \dfrac{n}{n^a} = n^{1-a}$ such subintervals.
- By combining subintervals corresponding to each of k variables y_j, we get $\sim (n^{1-a})^k = n^{(1-a) \cdot k}$ boxes.
- Each side of each box has size 2δ.

16. Proof (cont-d)

- Thus, each value y_j from this side differs from its midpoint $\widetilde{y}_j$ by no more that δ.
- So, by our choice of δ:
  - for each point $y = (y_1, \ldots, y_k)$ from the box,
  - the value $F(y_1, \ldots, y_k)$ differs from the value $F(\widetilde{y}_1, \ldots, \widetilde{y}_k)$ at the corresponding midpoint by no more than ε.
- Hence:
  - to find the desired range of the function f,
  - it is sufficient to find the values $F(\widetilde{y}_1, \ldots, \widetilde{y}_k)$ at the midpoints of all the boxes that have values y satisfying the constraints.
- Indeed, each value of f is ε-close to one of these midpoint values.
- Thus, the largest possible value of f is ε-close to the largest of these midpoint values.

17. Proof (cont-d)

- Similarly, the smallest possible of f is ε-close to the smallest of these midpoint values.
- So:
  - once we know which boxes are possible and which are not,
  - we will be able to compute both endpoints of the desired range with accuracy ε.
- There are no more than $\sim n^{(1-a)\cdot k}$ such midpoint values.
- Computing each value requires $\sim n^{p_c}$ time.
- So, once we determine which boxes are possible and which are not, we will need computation time $\sim n^{(1-a)\cdot k} \cdot n^{p_c} = n^{(1-a)\cdot k+p_c}$.
- How can we determine whether a box is possible?
- Each box $[y_1^-, y_1^+]\times\ldots\times[y_k^-, y_k^+]$ is determined by $2k$ linear inequalities $y_j^- \le y_j$ and $y_j \le y_j^+$, $j = 1, \ldots, k$.

18. Proof (cont-d)

- Substituting the expressions for y_j into these inequalities, and combining them with m linear constraints:
  - we get $2k + m$ linear constraints
  - that determine whether a box is possible.
- Namely:
  - if all these constraints can be satisfied, then the box is possible,
  - otherwise, the box is not possible.
- The problem of checking whether a system of linear constraints can be satisfied is known as linear programming.
- There exist feasible algorithms for solving this problem.
- For example, it can be solved in time $\sim (n + (2k + m)) \cdot n^{1.5}$.

19. Proof (cont-d)

- Checking this for all $\sim n^{(1-a)\cdot k}$ boxes requires time
$$\sim n^{(1-a)\cdot k} \cdot (n + 2k + m) \cdot n^{1.5}.$$
- The overall time for our algorithm consists of checking time and time for actual computation.
- It is thus bounded by time $\sim n^{(1-a)\cdot k} \cdot (n + m) \cdot n^{1.5} + n^{(1-a)\cdot k + p_c}$.
- This upper bound is polynomial in n and m.
- Thus our algorithm is indeed feasible.
- The proposition is proven.

20. 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).
- It was also supported 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.