How to Represent Uncertainty via Qudits: Probability Distributions, Regular, Intuitionistic, and Picture Fuzzy Sets, F-Transforms, etc.

Olga Kosheleva[1] and Vladik Kreinovich[2]

Departments of [1]Teacher Education and [2]Computer Science
University of Texas at El Paso, El Paso, Texas 79968, USA
olgak@utep.edu, vladik@utep.edu

1. Outline

- The need for faster computations necessitates the need to make computer components smaller and smaller.
- The smaller we make them, the more important is to take quantum effects into account.
- From this viewpoint, quantum computing – computing on devices for which we need to take quantum effects into account – is inevitable.
- Traditional quantum computing techniques are based on qubits – quantum analogues of 2-state components (bits).
- However, lately, it has been shown that it is often beneficial to use quantum analogues of d-state components for $d > 2$.
- Such analogues are known as *qudits*.

2. Outline (cont-d)

- Input to computations comes from measurements and expert estimates.
- In both cases, the values we submit to algorithms are known with uncertainty.
- In this talk, we analyze how different types of uncertainty can be represented in the qudit form.

3. We need faster computers

- Modern computers are very fast.
- However, there are still many practical problem for which their computation speed is not enough.
- An example of such a problem is tornado prediction.
- We can reasonably accurately predict tomorrow's weather – by spending several hours on a high-performance computer.
- We can also predict, by spending the same computation time, in what direction a tornado will turn in the next 15 minutes.
- For predicting weather, several hours of computing still result in a prediction being ahead of the actual event.
- However, for tornados, this computation time makes no sense: by the time we have our predictions, the tornado will already have turned.
- To solve such problems, we need faster computations.

4. To make computers faster, we need to make their components smaller

- The speed of current computers is limited – somewhat surprisingly – by fundamental physics.
- Namely, by the fact that, according to physics, no process can be faster than the speed of light 300 000 km/sec.
- The size of a usual laptop is approximately 30 cm size.
- The fastest way to send a signal from one of its sides to another one can be obtained if we divide 30 cm by the speed of light.
- As a result, we get one nanosecond – 10^{-9} of a second.
- During this time, a usual 4GHz computer – that performs 4 operations per nanosecond – will already perform 4 operations.
- From this viewpoint, the only way to drastically speed up computations is to drastically shrink the computer.
- Thus, we need to drastically shrink all its components.

5. Enter quantum effects

- The current computer cells are already almost the size of a few thousands of molecules.
- So if we drastically shrink them, their size will be comparable to a molecule size.
- To describe objects at such micro-size, it is no longer sufficient to use the usual Newton's mechanics.
- It is necessary to take into account effects of quantum physics – the physics of micro-world.
- This is exactly what is called *quantum computing*.
- It is computing with devices whose performance cannot be described without taking quantum effects into account.

6. Quantum computing: from necessary evil to spectacular (and scary) promises

- At first, computer engineers viewed these quantum effects purely negatively.
- For example, in quantum physics, results can only predicted with some probabilities.
- This is a big problem when we want to design a computer that returns the same (correct) answer every time.
- However, later, scientists learned how to "tame" these probabilities and come up with deterministic devices.
- Moreover, it turns out that in many cases, quantum effects the specific can speed up computations.
- E.g., quantum computing can find, in an unsorted n-element list, an element with a desired property in time proportional to $\sqrt{n}$.

7. Quantum computing: from necessary evil to spectacular (and scary) promises (cont-d)

- In non-quantum case, we cannot do it faster than in n computational steps.
- Indeed, if we do not check all n elements, we may miss the desired element.
- An even more drastic speedup is attained for the problem of representing a given integer as a product of prime numbers.
- For non-quantum computing the only available algorithms require computation time that grows exponentially with the number's length.
- Thus, for 100-digit numbers this time becomes larger than the lifetime of the Universe.
- With quantum computing, we can do it in feasible time.

8. Quantum computing: from necessary evil to spectacular (and scary) promises (cont-d)

- This example is very important, because:
  - all modern computer encryption algorithms – that make our communications private
  - are based on the difficulty of finding prime factors.
- So, when quantum computers will appear, all our supposedly secret messages will be available to everyone.

9. States in quantum physics

- A specific feature of quantum physics is that:
  - for every set of classical states $s, \ldots, s'$ – which in quantum physics are denoted as $|s\rangle, \ldots, |s'\rangle$
  - we can have *superpositions* of these states, i.e., states of the form

$$c_s|s\rangle + \ldots + c_{s'}|s'\rangle.$$

- Here $c_s, \ldots, c_{s'}$ are complex numbers for which $|c_s|^2 + \ldots + |c_{s'}|^2 = 1$.

10. Independent systems in quantum physics

- Suppose that we have two independent objects:
  - the first one with classical states s, $\ldots$, s', and
  - the second one with the states t, $\ldots$, t'.
- Suppose that the first object is in the state $c_s|s\rangle + \ldots + c_{s'}|s'\rangle$.
- Suppose that the second object is in the state $c'_t|t\rangle + \ldots + c'_{t'}|t'\rangle$.
- Then the system composed of these two objects is in the state

  $$c_s \cdot c'_t|s', t\rangle + \ldots + c_s \cdot c'_{t'}|s, t'\rangle + \ldots + c_{s'} \cdot c'_t|s, t\rangle + \ldots + c_{s'} \cdot c'_{t'}|s', t'\rangle.$$

- This joint state is called a *tensor product* of the states of these two systems.

11. Measurements in quantum physics

- In general, if in the superposition state $c_s|s\rangle + \ldots + c_{s'}|s'\rangle$, we measure the state of the system:
  - we will get $|s\rangle$ with probability $|c_s|^2$, ..., and
  - we will get $|s'\rangle$ with probability $|c_{s'}|^2$.
- The fact that we always get exactly one of these possible results implies that the sum of these probabilities should be equal to 1.
- This explains the above condition on the coefficients

$$c_s, \ldots, c_{s'}.$$

12. Enter qubits

- Most computers are based on the binary system.
- Its components are 2-state components corresponding to binary (0 or 1) digits known as *bits*.
- So naturally, most current quantum computing schemes are based on using quantum analogues of bits, known as *qubits*.
- In particular, since a bit has two states 0 and 1, a general state of a qubit is the state
$$c_0|0\rangle + c_1|1\rangle.$$
- Here c_0 and c_1 are complex numbers for which
$$|c_0|^2 + |c_1|^2 = 1.$$

13. Traditional approach to implementing qubits

- Because of the emphasis on qubits, to implement quantum computing, researchers:
  - find quantum systems that can be in several different classical states – e.g., ions, and
  - select one of these states as 0 and another one as 1.
- Ions and other physical quantum systems can be in many possible classical states.
- In this usual design of quantum computers, all other classical states are not used.

14. Enter qudits

- To further increase efficiency, a natural idea is thus to utilize these additional states; namely:
  - if we have d different states – which we can mark as states
  $$0, 1, \ldots, d-1,$$
  - then a general quantum state of this system has the form
  $$c_0|0\rangle + c_1|1\rangle + \ldots + c_{d-1}|d-1\rangle.$$
- Here $|c_0|^2 + |c_1|^2 + \ldots + |c_{d-1}|^2 = 1$.
- The states 0, 1, $\ldots$, $d-1$ can be naturally labeled by the d-base digits.
- Because of this labeling, the corresponding quantum states are called *quantum d-base digits*, or *qudits*, for short.
- It has been shown that the use of qudits can indeed further speed up quantum computations.

15. Need to represent uncertainty by qudits

- Input to computations comes from measurements and expert estimates.
- In both cases, the values we submit to algorithms are known with uncertainty.
- It is therefore desirable to represent the corresponding uncertainty in the form appropriate for quantum computing – i.e.:
  - by using qudits
  - or at least by using their particular case of qubits.
- We will show that many types of uncertainty information can indeed be naturally represented in this form.

16. Case of probabilistic uncertainty

- Let us start with the most traditional type of uncertainty – *probabilistic* uncertainty.
- In general, such an uncertainty means that:
  - we have several (n) alternatives – which we can denote by
  $$0, 1, \ldots, n-1,$$
  - and for each alternative i, we know its probability p_i.
- These probabilities should add up to 1: $p_0 + p_1 + \ldots + p_{n-1} = 1$.
- A natural qudit representation of this uncertainty means using a qudit with $d = n$ and taking $c_i = \sqrt{p_i}$.
- For these coefficients, the condition $|c_0|^2 + |c_1|^2 + \ldots = 1$ takes the form $p_0 + p_1 + \ldots = 1$ and is, thus, automatically satisfied.

17. Case of probabilistic uncertainty (cont-d)

- Suppose that we have such qudit representations of two independent probabilistic objects, with probabilities $p_0, \ldots, p_{n-1}$, and $q_0, \ldots, q_{m-1}$:

$$c_0|0\rangle + \ldots + c_{n-1}|n-1\rangle \text{ and } c'_0|0'\rangle + \ldots + c'_{m-1}|(m-1)'\rangle.$$

- Then, as one can easily see, the system consisting of these two objects is represented by the tensor product of these two qudit states.
- Indeed, for $c_i = \sqrt{p_i}$ and $c'_j = \sqrt{q_j}$, the probability $|c_i \cdot c'_j|^2$ of getting the state $|i, j\rangle$ is indeed equal to the independence-based value $p_i \cdot q_j$.

18. Case of fuzzy uncertainty: first idea

- In the *fuzzy* case, for each of n alternatives, we have a degree $\mu_i \in [0, 1]$ with the condition that $\max(\mu_0, \mu_1, \ldots, \mu_{n-1}) = 1$.
- In this case, there seems to be no sequence of numbers that adds to 1.
- To come up with such a sequence, we can use the fact – many times emphasized by Zadeh – that:
  - both fuzzy and probabilistic uncertainty can come from the same set of observations,
  - the only difference is in the normalization.
- In the probabilistic case, we normalize so that the sum is equal to 1.
- In the fuzzy case, we normalize so that the largest value is equal to 1.

19. Case of fuzzy uncertainty: first idea (cont-d)

- This way, we have a natural way to transform probabilities into fuzzy degrees and vice versa:
  - if we know the probabilities p_i, then normalization-to-maximum transforms these values into fuzzy degrees:

$$\mu_i = \frac{p_i}{\max(p_0, p_1, \ldots, p_{n-1})};$$

  - similarly, if we know fuzzy degrees μ_i, then normalization-to-sum transforms these values into probabilities:

$$p_i = \frac{\mu_i}{\mu_0 + \mu_1 + \ldots + \mu_{n-1}}.$$

- So, a natural way to use qudits to represent fuzzy information $\mu_0, \mu_1, \ldots$ is:
  - to transform this information into the probabilistic form, and then
  - use the above qudit-based representation of probabilities.

20. Case of fuzzy uncertainty: alternative idea

- An alternative idea is to use the fact that:
  - in the traditional fuzzy logic,
  - the degree d_+ to which a statement S is true and the degree d_- to which this statement is false add up to 1.
- Thus, we can represent such a pair of degrees by a qubit in which $c_0 = \sqrt{d_+}$ and $c_1 = \sqrt{d_-}$.

21. Case of intuitionistic fuzzy logic

- The alternative idea can also be naturally extended to *intuitionistic* fuzzy degrees, were $d_+ + d_- \le 1$.
- In this case, we have $d_+ + d_- + d_0 = 1$, where $d_0 \stackrel{\text{def}}{=} 1 - d_+ - d_-$ is the degree of indifference.
- To represent such degrees, it is reasonable to use 3-state qudit states $c_0|0\rangle + c_1|1\rangle + c_2|2\rangle$ with $c_0 = \sqrt{d_+}$, $c_1 = \sqrt{d_-}$, and $c_2 = \sqrt{d_0}$.
- Such a representation is even more natural in intuitionistic fuzzy logic of second type (also known as Pythagorean fuzzy logic).
- There, the degrees d_+ and d_- are related by a formula $d_+^2 + d_-^2 = 1$ (or $d_+^2 + d_-^2 + d_0^2 = 1$).
- In this case, we can take $c_0 = d_+$ and $c_1 = d_-$ (and $c_2 = d_0$).

22. Case of picture fuzzy logic

- A similar representation is possible for *picture fuzzy degrees* in which
$$d_+ + d_- + d_0 \le 1.$$
- In this case, $d_+ + d_- + d_0 + d_u = 1$, where $d_u \stackrel{\text{def}}{=} 1 - d_+ - d_- - d_0$ is the additional degree.
- To represent such degrees, it is reasonable to use 4-state qudit states $c_0|0\rangle + c_1|1\rangle + c_2|2\rangle + c_3|3\rangle$ with
$$c_0 = \sqrt{d_+},\quad c_1 = \sqrt{d_-},\quad c_2 = \sqrt{d_0},\ \text{and}\ c_3 = \sqrt{d_u}.$$

23. Case of F-transforms

- A useful notion of *F-transform* is based on families of membership functions $A_0(x), \ldots, A_{n-1}(x)$ for which, for each x, we have

$$A_0(x) + \ldots + A_{n-1}(x) = 1.$$

- Thus, for each x, we can describe the values of all n basic membership functions by using an n-state qudit with $c_i = \sqrt{A_i(x)}$.

24. Important comment

- In all these example, we can decrease the number of needed qudit states in half if:
  - instead of considering only real-valued coefficients c_i – as in the current quantum computing algorithms,
  - we allow general complex-valued coefficients $c_i = a_i + b_i \cdot \mathrm{i}$, where

$$\mathrm{i} \stackrel{\text{def}}{=} \sqrt{-1}.$$

- In these terms, since $|a + b \cdot \mathrm{i}|^2 = |a|^2 + |b|^2$, the condition that $|c_0|^2 + \ldots + |c_{d-1}|^2 = 1$ takes the form

$$|a_0|^2 + |b_0|^2 + \ldots + |a_{d-1}|^2 + |b_{d-1}|^2 = 1.$$

- So, e.g., to represent a general picture degree, it is sufficient to use a 2-state qubit $c_0|0\rangle + c_1|1\rangle = (a_0 + b_0 \cdot \mathrm{i})|0\rangle + (a_1 + b_1 \cdot \mathrm{i})|1\rangle$ with

$$a_0 = \sqrt{d_+}, \quad b_0 = \sqrt{d_-}, \quad a_1 = \sqrt{d_0}, \text{ and } b_1 = \sqrt{d_u}.$$

25. 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.
- The authors are greatly thankful to Krassimir Atanassov for his encouragement.