Uncertainty: Ideas Behind Neural Networks Lead Us Beyond KL-Decomposition and Interval Fields

Michael Beer[1], Olga Kosheleva[2], and Vladik Kreinovich[2]
[1]Institute for Risk and Reliability, Leibniz University Hannover
Hannover, Germany, beer@irz.uni-hannover.de
University of Texas at El Paso, El Paso, TX 79968, USA
{olgak,vladik}@utep.edu

Presentation at SSCI'2021

1. Dependencies are ubiquitous

- In a nutshell, the main purpose of engineering is to make the world better: whether:
  - by controlling natural phenomena (e.g., in damming flooding rivers) or
  - by coming up with devices (and ways to use them) that will make the world better for us.
- For this purpose, we need:
  - to describe the current state of world, and
  - to predict how the state of the world will change under our actions.
- Describing the state of world means describing:
  - at each spatial point (x, y) (or, in the 3-D case, (x, y, z)),
  - the values of all relevant quantities q at this point.
- In other words, we want to know the functions $q(x, y)$ or $q(x, y, z)$.

2. Dependencies are ubiquitous (cont-d)

- To decide how to change the state of the world, we need to know:
  - how the resulting state – i.e., how the resulting values of different quantities
  - will depend on the parameters $a_1, \ldots, a_n$ characterizing possible actions.
- In other words, we want to know the dependence $q(a_1, \ldots, a_n)$.
- In all these cases, we want to known a function – or functions.

3. Need to take uncertainty into account

- Information about dependencies comes from measurements.
- Measurements are never absolutely accurate.
- So, we never know the exact function $q(a_1, \ldots, a_n)$.
- There is a whole class $\mathcal{F}$ of different functions which are all consistent with the known knowledge.
- In many cases:
  - in addition to knowing this set $\mathcal{F}$,
  - we also know the frequencies (= probabilities) with which different functions from $\mathcal{F}$ occur in similar situations.
- In mathematical terms, this situation is known as a *random field*, or, in the 1-D case, a *random process*.

4. Need to simulate

- What can we do when the equations are too complicated to solve explicitly?
- A natural idea is to use computer simulation of the corresponding processes.
- To make simulations realistic, we need to take uncertainty into account.
- For this purpose, it is desirable to come up with simulations that would produce different functions from the corresponding class $\mathcal{F}$; also:
  - if we know the frequencies (probabilities) of different functions,
  - we want to produce different functions from $\mathcal{F}$ with exactly these frequencies.

5. What can be the input to such simulations

- How can we simulate different functions from the given class?
- A reasonable idea is to use simplest 1-D type of the corresponding uncertainty as inputs.
- Sometimes, we do not know the probabilities, we only know the class (= set) F.
- In this case, a natural idea is to use numbers for which we also know only they belong to some set of real numbers.
- The simplest such set is an interval $[-1, 1]$.
- So, a natural idea is to use, as inputs, numbers $c_1, \ldots, c_N$ from the interval $[-1, 1]$.

6. What can be the input to such simulations (cont-d)

- In situations when we know the probabilities, a reasonable idea is to use simple independent 1-D random variables c_i.
- E.g., uniformly distributed on the interval $[0, 1]$ or normally distributed with mean 0 and standard deviation 1.
- So, we need algorithms that would transform the values $c = (c_1, \ldots, c_N)$ into the corresponding functions $f_c(a_1, \ldots, a_m) \in \mathcal{F}$.

7. Problem: we need feasible-to-compute algorithms

- The dependencies from the class $\mathcal{F}$ are often rather complex.
- Their computation is very time-consuming even if we ignore the uncertainty.
- For example, accurately predicting tomorrow's weather requires spending several hours on a high-performance compute.
- Taking uncertainty into account would make the computational problems:
  - even more complex and thus,
  - requiring even more computation time.
- It is therefore desirable to come up with the fastest possible simulation algorithms.

8. What we do in this talk

- We show that:
  - the main ideas behind neural networks help explain
  - why KL-decomposition and interval fields are efficient in uncertainty-related simulations.
- We also show that these ideas help to go beyond these techniques.
- And we explain why we need to go beyond these two techniques.

9. What elementary steps can we use to speed up computations

- Every computation is composed of elementary steps.
- To speed up computations, a reasonable idea is to perform several computational steps in parallel, on different processors.
- This is how high-performance computers work.
- The overall computation time is the sum of computation times of different computational steps (performed in parallel).
- Thus, to speed up computations, we need:
  - to make these steps as fast as possible, and
  - to minimize the overall number of such steps.

10. What are the fastest possible steps?

- On each computational step, we transform some inputs $x_1, \ldots, x_n$ into one or more outputs y.
- The output y is usually uniquely determined by the inputs.
- So, in mathematical terms, we compute the value $f(x_1, \ldots, x_n)$ of an appropriate function.
- Which functions are the easiest to compute?
- In general, functions can be linear and non-linear.
- Clearly, linear functions $y = w_0 + w_1 \cdot x_1 + \ldots + w_n \cdot x_n$ are the easiest (and thus, the fastest) to compute.
- So linear functions must be among the corresponding elementary computational steps.

11. Need for nonlinear steps

- We cannot have only linear steps.
- If we only use linear steps, then what we will compute is a composition of linear functions – and this composition is always linear.
- On the other hand, many real-life processes are non-linear.
- Thus, we also need to use non-linear elementary computational steps.
- In general:
  - the more inputs we use,
  - the more time is needed for computations.
- Thus, the fastest to compute are the nonlinear functions that have the smallest possible number of inputs – only one. So:
  - in additional linear elementary computational steps,
  - we should also use step that consist of applying a nonlinear function $y = f(x)$ to a single input.

12. Layers

- Computations on a parallel computer comes in what is called layers:
  - first, all the processors perform one type of operations,
  - then they perform other types of operations, etc.
- As we have argued, each layer must consist of:
  - either computing linear functions,
  - or computing non-linear functions of one variable.
- Let us denote:
  - a linear layer by L, and
  - a nonlinear layer by NL.

13. Layers must interleave

- Our goal is to speed up computations.
- So, it does not make sense to have a linear layer following a linear layer.
- Indeed, in such a $L-L$ configuration, all we are computing are compositions of linear functions – which are also linear.
- Thus, we could as well compute these functions faster, by using a single linear layer instead of two.
- Similarly, it does not make sense to have a nonlinear layer following a nonlinear layer.
- Indeed, in such a configuration, all we are computing are compositions of functions of one variable, i.e., expressions $y = f(g(x))$.
- These expressions are also simply functions of one variable.
- Thus, we could as well compute these functions faster, by using a single nonlinear layer instead of two.

14. Layers must interleave (cont-d)

- So, layers must interleave:
  - a linear layer (which is not final) must be followed by a nonlinear layer, and
  - a nonlinear layer (which is not final) must be followed by a linear layer.

15. How many layers do we need?

- To speed up computations, we need to use the smallest possible number of layers.
- With one layer, we can compute either linear functions, or nonlinear functions of a single input.
- In practice, however, many real-life dependencies are nonlinear, and have more than one input.
- Thus, one layer is not enough.
- In the case of two layers, we can have:
  - either a linear layer followed by a nonlinear layer,
  - or a nonlinear layer followed by a linear layer.
- Let us show that in both cases, we cannot cover all possible dependencies.
- Specifically, we will show that we are not even able to cover the product $y = x_1 \cdot x_2$.

16. $NL - L$ configuration

- In the first layer of $NL - L$ configuration, we compute the values $f_i(x_1)$ and/or $g_j(x_2)$.
- In the second layer, we compute a linear combination of these results, i.e., the value $y = F_1(x_1) + F_2(x_2)$.
- If we could have $x_1 \cdot x_2 = F_1(x_1) + F_2(x_2)$, then:
  - from the fact that $0 \cdot x_2 = 0$ for all x_2, we conclude that $0 = F_1(0) + F_2(x_2)$, i.e., that $F_2(x_2) = -F_1(0) = \text{const}$;
  - similarly, from the fact that $x_1 \cdot 0 = 0$ for all x_1, we conclude that $0 = F_1(x_1) + F_2(0)$, i.e., that $F_1(x_1) = -F_2(0) = \text{const}$.
- Thus, the sum $F_1(x_1) + F_2(x_2)$ of two constant functions is also a constant, not depending on x_i.
- So, it cannot thus be equal to the product $x_1 \cdot x_2$.

17. $L - NL$ configuration

- In the first layer, we compute a linear combination $w_0+w_1\cdot x_1+w_2\cdot x_2$.
- In the second layer, we apply a nonlinear function $f(x)$, resulting in
$$F(x_1,x_2)=f(w_0+w_1\cdot x_1+w_2\cdot x_2).$$
- Let us show that the product $x_1\cdot x_2$ cannot be equal to such an expression.
- Indeed, in the case of such an equality, we must have $w_1\neq 0$ and $w_2\neq 0$.
- Otherwise the expression $F(x_1,x_2)$ will not depend on x_1 or on x_2.
- Now, for each x_2, we have $0=0\cdot x_2=F(w_0+w_2\cdot x_2)$.
- Since $w_2\neq 0$, the expression $w_0+w_2\cdot x_2$ can take any real value x, so we have $F(x)=0$ for all x.
- Therefore, it is not possible to have $0=F(1\cdot 1)=1\cdot 1$.
- So, 2 layers are not enough, we must have at least 3 layers.

18. Which 3-layer configuration is the fastest

- Since layers must interleave, we can have two possible 3-layer configurations: $L - NL - L$ and $NL - L - NL$.
- In the first case, we have one nonlinear layer and two linear layers.
- In the second case, we have two nonlinear layers and one linear layer.
- Since a linear layer is faster than a nonlinear one, the $L - NL - L$ configuration is faster.
- In the first later, each processor k computes a linear combination of inputs: $z_k = w_{k0} + w_{k1} \cdot x_1 + \ldots + w_{kn} \cdot x_n$.
- In the second layer, we apply some nonlinear function of each outputs y_k of the first layer, computing $y_k = f_k(y_k)$ for some functions $f_k(x)$.
- Finally, on the last linear layer we compute a linear combination of the values y_k: $y = W_0 + W_1 \cdot y_1 + \ldots + W_K \cdot y_K$.
- This is what the traditional neural networks compute.

19. Three layers are sufficient

- 3-layer configuration is sufficient to represent any continuous function on bounded domain with any desired accuracy; indeed:
  - each such function can be represented as a Fourier transform,
  - i.e., as a linear combination of sines of linear combinations of the inputs.
- This corresponds to the above expression for $f_1(x) = \ldots = f_K(x) = \sin(x)$.
- The use of neural networks (NN) has indeed led to much faster algorithms for processing uncertainty.
- For realistically complex problems, NN require high-performance computers to finish computations in reasonable time.
- Thus, a further computational speed up is desirable.

20. Main Ideas Behind Neural Networks Explain KL-Decomposition and Interval Fields

- We want to have a fast algorithm that would:
  - transform values $c = (c_1, \ldots, c_N)$ that represent simple uncertainties – interval or random
  - into the corresponding values $f_c(a_1, \ldots, a_n)$.
- As we have mentioned, the fastest possible algorithms are linear.
- Thus, we arrive at the following expression which is linear in c_i:

$$w_0(a_1, \ldots, a_n) + c_1 \cdot w_1(a_1, \ldots, a_n) + \ldots + c_N \cdot w_N(a_1, \ldots, a_n).$$

- When each c_i is taking values from the interval $[-1, 1]$, this expression becomes a particular case of so-called *interval fields*.
- This is indeed a useful techniques for analyzing such uncertainty.

21. Main Ideas Behind Neural Networks Explain KL-Decomposition and Interval Fields (cont-d)

- In general, an interval field is defined as the class of all such functions when each c_i are in a given interval $[\underline{c}_i, \overline{c}_i]$.
- It can be shown that this general definition can be reduced to the above case, when all the values come from the interval $[-1, 1]$.
- When c_i are independent random variables, a particular case of the expression is Karhunen-Loéve (KL) representation of a random field.
- Thus, the main ideas behind neural networks indeed explain the empirical success of KL-decomposition and interval field techniques.

22. Need to Go Beyond KL-Representations

- In the probabilistic case, we get a sum of a large number of independent random variables $c_i \cdot w_i(a_1, \ldots, a_n)$.
- In general, each of these terms is reasonably small.
- Otherwise, if a few of these terms were domineering, we could have left only these terms and ignore the others.
- It is known that:
  - under reasonable conditions,
  - the distribution of the sum of a large number of small independent random variable is close to Gaussian.
- This follows from the Central Limit Theorem.
- Thus, the above expression only works for Gaussian random fields – this is how KL representation is usually used.
- To describe such non-Gaussian distributions, we need to go beyond KL-representations.

23. Need to go beyond interval fields

- Each segment $[-1, 1] \cdot w_i(a_1, \ldots, a_n)$ is a convex set.
- It is known the Minkowski sum of convex sets is a convex set.
- Thus, only convex sets of functions can be represented in this form.
- Not all sets of possible values are convex. Indeed:
  - it is known that a nonlinear transformation, in general, transforms convex sets into non-convex ones, and,
  - as we have mentioned, many real-life transformations are nonlinear.

24. How to Go Beyond KL-Decomposition and Interval Fields

- We showed that KL-decomposition and interval fields correspond to using just one linear layer.
- In many practical situations, we need to go beyond these techniques.
- We therefore need to turn to the next fastest approach, when we have two layers.
- Since the layers must interleave, we have two possible 2-layer configurations: $NL-L$ and $L-NL$.

25. $NL - L$ approach

- For our problem, $NL - L$ approach means that:
  - we first apply some nonlinear transformations $c_i \mapsto f_i(c_i)$ to the inputs c_i, and
  - then perform a linear transformation:

$$w_0(a_1, \ldots, a_n) + f_1(c_1) \cdot w_1(a_1, \ldots, a_n) + \ldots + f_N(c_N) \cdot w_N(a_1, \ldots, a_n).$$

- Will this help? Not really.
- In the probabilistic case, the variables $f_i(c_i)$ are still independent.
- So the same Central Limit Theorem still shows that, as a result, we get a Gaussian field or a Gaussian process.
- In the interval case, for continuous functions $f_i(c_i)$, the range of this function when $c_i \in [-1, 1]$ is still an interval.
- So we still get an interval field.

26. $L - NL$ approach

- In this approach, we apply a nonlinear function $f(x)$ to the result of linear processing:

$$f_c(a_1, \ldots, a_m) = f\left(w_0(a_1, \ldots, a_n) + \sum_{i=1}^{N} c_i \cdot w_i(a_1, \ldots, a_n)\right).$$

- Empirical results show that this formula indeed leads to a very good description of non-Gaussian probabilistic uncertainty.
- We hope that it will be as useful in describe non-convex classes of functions.

27. What next?

- We have mentioned that the $L - NL$ approach cannot describe all possible dependencies.
- So, at some point, we will encounter practical situations when this approach needs to be replaced by a more accurate one.
- How can we do it?
- A natural idea is to consider 3-layer $L - NL - L$ approach:

$$f_c(a_1, \ldots, a_m) = W_0 + W_1 \cdot f_1 \left(w_{10}(a_1, \ldots, a_n) + \sum_{i=1}^{N} c_i \cdot w_{1i}(a_1, \ldots, a_n) \right) + \ldots +$$

$$W_K \cdot f_K \left(w_{K0}(a_1, \ldots, a_n) + \sum_{i=1}^{N} c_i \cdot w_{Ki}(a_1, \ldots, a_n) \right).$$

28. What next (cont-d)

- The $L-NL-L$ approach already have the universal approximation property.
- So it can describe, with any desired accuracy:
  - the corresponding class of functions,
  - and, if appropriate, the probability distribution on this class of functions.

29. 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.