For Which Activation Functions, Any Neural Network Is Equivalent to a Takagi-Sugeno Fuzzy System with Constant or Linear Outputs?

Barnabas Bede[1], Olga Kosheleva[2], and Vladik Kreinovich[3]

[1]DigiPen Institute of Technology, 9931 Willows Rd.,
Redmond, WA 98052, USA, bbede@digipen.edu
[2,3]Departments of [2]Teacher Education and [3]Computer Science
University of Texas at El Paso,
500 W. University, El Paso, Texas 79968, USA
olgak@utep.edu, vladik@utep.edu

1. Need for explainable AI

- Many recent results of using deep neural networks are spectacular.
- However, there is a problem: these results are often not explainable.
- This is important in social applications.
- E.g., when the neural network is used to decide whether to give a loan, whether to apply some treatment to a patient, etc.
- The need for an explanation comes from the fact that the neural networks are not perfect, they sometimes produce wrong answers.
- Of course, human decision makers are also not perfect.
- However, with a human decision maker, you can always ask for reasons for his/her decision.
- Thus, we can check how convincing these reasons are – and, based on this, filter out some incorrect decisions.
- For a neural system, usually no reasons are provided, so it is not easy to filter out wrong decisions.

2. Translation into fuzzy as a possible first step towards explainability

- Explainability means to be able to describe the decision in human-understandable, natural-language terms.
- So, to achieve explainability, a natural idea is to look for existing techniques that relate:
  - natural-language explanations with
  - precise computer-based decisions.
- This immediately brings us into the realm of fuzzy techniques.
- Indeed, these techniques were specifically designed:
  - to translate natural-language expert recommendations
  - into precise computer-understandable form.
- Of course, translation into natural language does not necessarily mean that we already have a convincing explanation.
- However, in general, this may be a first step towards an explanation.

3. Question that we deal with in this talk

- There are many different versions of fuzzy techniques and many different types of neural networks.
- A neural network usually consists of *neurons* each of which:
  - takes values $x_1, \ldots, x_n$ and
  - produces a value $y = s(a_0 + a_1 \cdot x_1 + \ldots + a_n \cdot x_n)$.
- Here, a_i are numerical coefficients that are determined during the network's training.
- The function $s(z)$ is a continuous function known as an *activation function.*
- Some neurons process the inputs $v = (v_1, \ldots, v_m)$ to the network.
- Other neurons process the outputs of previously active neurons.
- One of the outputs of one of the neurons is then returned to the user as the computation result V.

4. Question that we deal with in this talk (cont-d)

- Traditional neurons use the sigmoid function $s(z) = 1/(1+\exp(-z))$.
- Most current networks use ReLU function $s(z) = \max(0, z)$.
- Many other activation functions have been proposed and effectively used.
- As fuzzy techniques, we will use one of most widely used versions: Takagi-Sugeno techniques.
- In this technique, a function $V = f(v_1, \ldots, v_m)$ is characterized by rules of the type

$$\text{if } m_i(v) \text{ then } f_i(v).$$

- Here $0 \le m_i(v) \le 1$ for all i and v and the functions $f_i(v)$ are usually either constants or linear functions.
- The function computed by this technique is $f(v) = \dfrac{\sum\limits_i m_i(v) \cdot f_i(v)}{\sum\limits_i m_i(v)}$.

5. Question that we deal with in this talk (cont-d)

- A natural question is: for what activated functions the following is true:
  - every function computed by the corresponding neural network
  - can also be computed by a TS system?
- In this talk, we provide an answer to this question.

6. What if we use Takagi-Sugeno systems with constant outputs: result

For each function $s(z)$, the following two conditions are equivalent to each other:

- *the function $s(z)$ is bounded, i.e., there exists a bound B such that $|s(z)| \le B$ for all z, and*
- *every function computed by a network of neurons with this activation function can be computed by a TS system with constant outputs.*

7. Proof

- The value V computed by a TS system is a convex combination of the values $f_i(v)$.
- So, when all the outputs $f_i(v)$ are constants $f_i(v) = f_i$, all the values V are bounded by the largest of the absolute values $|f_i|$.
- Hence, every function computed by a TS system with constant outputs is bounded.
- Thus:
  - if a function $s(z)$ is not bounded,
  - then this same function – computed by a single neuron – cannot be computed by a TS system with constant outputs.

8. Proof (cont-d)

- So, to complete the proof, it is sufficient to prove that:
  - if the activation function *is* bounded,
  - then any function computed by the corresponding neural network can also be computed by a TS system with constant outputs.
- Indeed, the value $V = f(v)$ computed by a neural network comes from one of the neurons.
- It is, thus, bounded by the bound B: $-B \le f(v) \le B$.
- So, to compute this function, we can use the following two Takagi-Sugeno rules:

$$\text{if } \frac{f(v)+B}{2B} \text{ then } B; \quad \text{if } \frac{B-f(v)}{2B} \text{ then } -B.$$

- In this case, the sum of the two functions $m_i(v)$ is 1:

$$\frac{f(v)+B}{2B} + \frac{B-f(v)}{2B} = \frac{2B}{2B} = 1.$$

9. Proof (cont-d)

- So the result V of this system is simply equal to

$$m_1(v) \cdot f_1(v) + m_2(v) \cdot f_2(v) = \frac{f(v) + B}{2B} \cdot B + \frac{B - f(v)}{2B} \cdot (-B) =$$

$$\frac{f(v) + B}{2} - \frac{B - f(v)}{2} = \frac{f(v) + B - B + f(v)}{2} = \frac{2f(v)}{2} = f(v).$$

- The proposition is proven.

10. Discussion

- A similar result – with the same proof – holds if we consider a neural network in which:
  - different neurons
  - can have different activation functions.
- The only condition we need is that all these activation functions are bounded.
- **Proposition.** *For each neural network in which all activation functions are bounded:*
  - *every function computed by this network*
  - *can also be computed by a Takagi-Sugeno system with constant outputs.*
- A similar argument shows that if we limited ourselves to a bounded domain D of values v.

11. Discussion (cont-d)

- Then the result of any neural network, with any activation function, can be computed by a Takagi-Sugeno system with bounded outcomes.
- **Proposition.**
  - *Let D be a bounded domain.*
  - *Let there be a neural network with any activation functions that computes some function $f(v)$.*
  - *Then there exists a Takagi-Sugeno system with constant outputs that computes $f(v)$ for all v from the domain D.*

12. Proof

- By definition, the function computed by a neural network is a composition of functions computed by individual neurons.
- Since all activation functions are continuous, the function computed by each neuron is continuous.
- Thus, the overall function computed by a neural network is continuous.
- A continuous function on a bounded domain is always bounded.
- Thus, we can use the construction from the proof of the first Proposition.

13. What if we use Takagi-Sugeno systems with linear outputs

- We say that a function $f(x_1, \ldots, x_n)$ is *linearly bounded* if there exist positive coefficients $c_0, c_1, \ldots, c_n$ for which we always have

$$|f(x_1, \ldots, x_n)| \le c_0 + c_1 \cdot |x_1| + \ldots + c_n \cdot |x_n|.$$

- For example, every linear function is linearly bounded, as well as each function computed by a single ReLU neuron.

- **Proposition.** *For each function $s(z)$, the following two conditions are equivalent to each other:*

  – *the function $s(z)$ is linearly bounded, and*

  – *every function computed by a network of neurons with this activation function can be computed by a TS system with linear outputs.*

14. Proof

- Since all linear outputs are linearly bounded, one can easily check that their convex combination V is also linearly bounded.
- So, if an activation function is not linearly bounded, it cannot be computed by such a Takagi-Sugeno system.
- So, to prove this result, it is sufficient to prove that:
  - every function computed by a neural network – consisting of neurons with linearly bounded neurons
  - can be computed by a Takagi-Sugeno system with linear outputs.
- It is easy to prove that the composition of linearly bounded functions is also linearly bounded.
- So, the function $f(v)$ computed by a neural network is also linearly bounded: $|f(v)| \leq c_0 + c_1 \cdot |v_1| + \ldots + c_m \cdot |v_m|$ for some $c_i > 0$.

15. Proof (cont-d)

- To proceed, we will use the following simple result: that:
  - if $|x| \leq a + b$ for some $a > 0$ and $b > 0$,
  - then we can represent x as $x = x_a + x_b$, where $|x_a| \leq a$, $|x_b| \leq b$,
  - and for fixed a and b, both x_a and x_b continuously depend on c.
- Indeed, let us take, as x_a, the closest to x value from the interval $[-a, a]$, i.e.:
  - if $-a \leq x \leq a$, we take $x_a = x$;
  - if $x > a$, we take $x_a = a$; and
  - if $x < -a$, we take $x_a = -a$.
- In all three cases, we can check that for $x_b = x - x_a$, we have $|x_b| \leq b$.
- In the first case, $x_b = 0$, so this inequality is clearly satisfied.
- In the second case, we get $x_b = x - a$.
- From $x \leq a + b$, we conclude that $x - a \leq b$, i.e., indeed, $x_b \leq b$.

16. Proof (cont-d)

- Since $x > a$, we have $x_b = x - a > 0$ and thus, indeed, $x_b \geq -b$.
- The third case can be proved similarly.
- If we have $|x| \leq a_1 + \ldots + a_k$ for some $a_i > 0$, then:
  - by applying the above simple result first,
  - we can conclude that $x = x_1 + x_{-1}$, where $|x_1| \leq a_1$ and

$$|x_{-1}| \leq a_2 + \ldots + a_k.$$

- By applying this result again, this time to x_{-1}, we can conclude that $x_{-1} = x_2 + z_{-2}$, where $|x_2| \leq a_2$ and $|x_{-2}| \leq a_3 + \ldots$, etc.
- After $k - 1$ steps, we conclude that $x = x_1 + \ldots + x_k$, where $|x_i| \leq a_i$.
- Let us apply this result to the inequality

$$|f(v)| \leq c_0 + c_1 \cdot |v_1| + \ldots + c_m \cdot |v_m|.$$

- Then, we conclude that the function $f(v)$ is equal to the sum $f(v) = F_0(v) + F_1(v) + \ldots + F_m(v)$, where $|F_0(v)| \leq c_0$ and $|F_i(v)| \leq c_i \cdot |v_i|$.

17. Proof (cont-d)

- Then, we can represent this function by using the following rules:

  "if $m_i^{\pm}(v)$ then $f_i^{\pm}(v)$" and "if $1/(m+1) - m_i^+(v) - m_i^-(v)$ then 0",

- Here:

$$m_0^+(v) = \frac{1}{m+1} \cdot \max\left(0, \frac{F_0(v)}{c_0}\right), \quad m_0^-(v) = \frac{1}{m+1} \cdot \max\left(0, -\frac{F_0(v)}{c_0}\right),$$

$$m_i^+(v) = \frac{1}{m+1} \cdot \max\left(0, \frac{F_i(v)}{c_i \cdot v_i}\right), \quad m_i^-(v) = \frac{1}{m+1} \cdot \max\left(0, -\frac{F_i(v)}{c_i \cdot v_i}\right),$$

$$f_0^+(v) = (m+1) \cdot c_0, \quad f_0^-(v) = -(m+1) \cdot c_0,$$

$$f_i^+(v) = (m+1) \cdot c_i \cdot v_i, \quad f_-^-(v) = -(m+1) \cdot c_i \cdot v_i.$$

- In this case, the sum of all the functions $m_i(x)$ is 1, so the formula for TS turns into a simple sum of products.

- Let us show that for each i, the sum of the corresponding product terms in the formula is equal to $F_i(v)$.

18. Proof (cont-d)

- This will guarantee that the sum of all the terms corresponding to all i is indeed equal to the desired function $f(v)$.
- Indeed, when $F_i(v)/(c_i \cdot v_i) > 0$, then we have

$$m_i^+(v) \cdot f_i^+(v) = \frac{1}{m+1} \cdot \frac{F_i(v)}{c_i \cdot v_i} \cdot (m+1) \cdot c_i \cdot v_i = F_i(v).$$

- In this case, two other products corresponding to i are 0s:
  - the second because $m_i^-(v) = 0$ and
  - the third because the corresponding output function is 0.
- The proof for the case when $F_i(v)/(c_i \cdot v_i) < 0$ is similar.
- The proposition is proven.

19. Discussion

- To represent each function, we used $3m + 3$ rules – a very feasible amount.
- We could get slightly fewer rules, namely, $3m+2$, if, to describe $F_0(v)$, we would use a construction from the first Proposition.
- A similar result – with the same proof – holds if we consider a neural network in which:
  - different neurons
  - can have different activation functions.
- The only condition is that all these activation functions are linearly bounded.
- **Proposition.** *Suppose that we have a neural network in which all activation functions are linearly bounded; then:*
  - *every function computed by this network*
  - *can be computed by a Takagi-Sugeno system with linear outputs.*

20. Discussion (cont-d)

- These results cannot be directly extended to the case when an activation function is:
  - quadratically bounded,
  - i.e., bounded by some quadratic function of the inputs.
- Indeed, in this case, functions computed by Takagi-Sugeno systems are still quadratically bounded.
- However, a composition of two quadratic neurons, with $s(z) = z^2$, already computes a function z^4.
- The function z^4 grows faster than any quadratic function.

21. Discussion (cont-d)

- Thus, the function z^4 cannot be computed by such Takagi-Sugeno systems.
- We can, however, get a similar result:
  - if we use hierarchical Takagi-Sugeno systems,
  - in which the result of one such system serves as an input to other systems.

22. Acknowledgments

This work was supported in part:

- by the US National Science Foundation grants:
  - 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science),
  - HRD-1834620 and HRD-2034030 (CAHSI Includes),
  - EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES),
- by the AT&T Fellowship in Information Technology, and
- by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) Focus Program SPP 100+ 2388, Grant Nr. 501624329,