12-2014

Need for Data Processing Naturally Leads to Fuzzy Logic (and Neural Networks): Fuzzy Beyond Experts and Beyond Probabilities

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Hung T. Nguyen
New Mexico State University - Main Campus, hunguyen@nmsu.edu

Songsak Sriboonchitta
Chiang Mai University, songsakecon@gmail.com

Comments:
Technical Report: UTEP-CS-14-79

Need for Data Processing Naturally Leads to Fuzzy Logic (and Neural Networks): Fuzzy Beyond Experts and Beyond Probabilities

Vladik Kreinovich[1], Hung T. Nguyen[2,3], and
Songsak Sriboonchitta[3]
[1]Department of Computer Science
University of Texas at El Paso
500 W. University, El Paso, TX 79968, USA
email vladik@utep.edu
[2]Department of Mathematical Sciences
New Mexico State University
Las Cruces, New Mexico 88003, USA
email hunguyen@nmsu.edu
[3]Faculty of Economics, Chiang Mai University
Chiang Mai, Thailand, email songsakecon@gmail.com

Abstract

Fuzzy techniques have been originally designed to describe imprecise ("fuzzy") expert knowledge. Somewhat surprisingly, fuzzy techniques have also been successfully used in situations without expert knowledge, when all we have is data. In this paper, we explain this surprising phenomenon by showing that the need for optimal processing of data (including crisp data) naturally leads to fuzzy and neural data processing techniques.

This result shows the potential of fuzzy data processing. To maximally utilize this potential, we need to provide an operational meaning of the corresponding fuzzy degrees. We show that such a meaning can be extracted from the above justification of fuzzy techniques. It turns out that, in contrast to probabilistic uncertainty, the natural operational meaning of fuzzy degrees is indirect – similarly to the operational meaning of geometry and physics in General Relativity.

1 Formulation of the Problem: a Search for Explanation of Fuzzy Techniques' Success and a Related Search for the Meaning of Fuzzy Degrees

Need for processing imprecise ("fuzzy") expert knowledge. In many areas of human activity – medicine, cooking, arts, sports – there are some individuals who are much better than others. In each subarea of medicine, there are a few skilled doctors who diagnose and treat the corresponding diseases better than others. There are few skilled chefs whose dishes taste much better, drivers whose driving is better – is smoother and/or saves more fuel, etc.

It is not possible for the few best doctors to treat all the patients, for the few best chefs to cook for all the restaurants, and for the best drivers to drive all the buses. It is therefore desirable to incorporate the skill of the best experts into an automated computer-based system that would help others. In other words, we would like to have an expert system that helps medical doctors diagnose and treat diseases, helps drivers drive cars, etc. A natural idea is to elicit the corresponding knowledge from the drivers. The problem is that this knowledge rarely comes in precise computer-understandable form. A skilled driver, when asked about a specific road situation, does not say "and then I press the brakes with the force of 1.0 Newtons for 0.6 seconds", the driver usually says something like "and I brake a little bit for a short period of time, to adjust my speed". To incorporate such expert knowledge into a computer-based system, we therefore need to describe the corresponding imprecise ("fuzzy") terms like "a little bit" and "short" in precise computer-understandable terms.

Fuzzy techniques as a way of describing imprecise expert knowledge. Sometimes, the expert statements are precise. For example, an expert driver may say that if the car's speed accidentally goes above the speed limit (e.g., 100 km/h), the driver should brake. The condition of this rule is very clear ("crisp"): it is satisfied for any velocity $v > 100$ and it is not satisfied for any velocity $v < 100$. The corresponding condition $v > 100$ is either true or false, which, in the computer, is usually represented as 1 or 0.

In contrast, words like "short" often have no precise meaning. The expert may say that 0.1 sec is for sure short, while 3.0 sec is for sure not short, but he may be hesitant about intermediate values like 1.0 sec. To describe such hesitancy, Lotfi Zadeh proposed, in his pioneer paper [15], to characterize a statement like "1.0 sec is short" by a value between 0 and 1 describing the degree to which, to the expert, 1.0 sec is short. This idea formed the basis of *fuzzy logic*.

The introduction of such degrees necessitates other changes. For example, a condition for an action is sometimes a propositional combination of different statements. For example, an expert may have a special rule describing what to do if the nearby car is close *and* starts to brake hard. For precise statements, it is easy to determine the truth value of the "and"-combination if we know

the truth values of both component statement. It is therefore desirable, in the fuzzy case, to similarly estimate the degree to which the "and"-statement is true, based on the degrees to which both original statements hold. For that, we need to extend the "and"- (and "or"-) operations from the 2-valued set $\{0, 1\}$ of the classical logic to the whole interval $[0, 1]$. In [15], Zadeh proposed to use the simplest such extensions $\min(a, b)$ and $\max(a, b)$ – which remain among the most widely used fuzzy operations, as well as more complex functions, such as $a \cdot b$ and $a + b - a \cdot b$.

As a result of applying these "and"- and "or"-operations to the original expert rules, we get different possible conclusions and recommendations x, each with its own degree of confidence $\mu(x)$. These recommendations can serve as the desired advice and help for other medical doctors, drivers, etc. In some practical situations, however, we need to go beyond that: for example, if we use this scheme in an automatic controller, we need to transform such a fuzzy conclusion $\mu(x)$ into a single value $\overline{x}$. This operation is known as *defuzzification* [5, 11]. The most widely used is *centroid* defuzzification $\overline{x} = \dfrac{\int x \cdot \mu(x)\, dx}{\int \mu(x)\, dx}$. In the discrete case, when we have finitely many values x_i with degrees $\mu(x_i)$, we can use a discrete version of this defuzzification formula $\overline{x} = \dfrac{\sum\limits_i x_i \cdot \mu(x_i)}{\sum\limits_i \mu(x_i)}$.

Comment. It should be mentioned that centroid does not always work: e.g., in a situation when a car encounters an obstacle on an empty road, we can go around this obstacle either from the left, or from the right. When both situations are equally possible, defuzzification of the corresponding function $\mu(x)$ of a turn angle x will lead us, due to symmetry, to the value $\overline{x} = 0$, that will lead us heads-on into the obstacle :-(

Successes and challenges. Fuzzy techniques have led to many successful applications, especially in intelligent control; see, e.g., [5, 11] and references therein. However, there remain two main challenges.

The first challenge is that while fuzzy techniques have been invented to describe imprecise expert knowledge, they are also often successful in situations when no expert knowledge is present, when all we have is pure data. The second challenge is that while, e.g., subjective probabilities have a precise operational meaning – and thus, can be extracted from an expert by appropriate elicitation procedures – there is no clear operational meaning of fuzzy techniques. As a result, different experts may describe the same behavior by using different degrees – and, moreover, some experts cannot meaningfully provide such degrees at all.

What we do in this paper. In this paper, we show that the need for optimal data processing naturally leads to fuzzy (and neural) techniques – which explains the first challenge, and that this explanation naturally leads to an operational meaning of fuzzy degrees.

This paper is aimed both at specialists in fuzzy as well as at the general

audience, ranging from sceptics – who we hope to convince that the use of fuzzy techniques makes perfect sense – to many practitioners who use fuzzy techniques without a good understanding of why and how these techniques work – to such practitioners, prior empirical success is a good reason to use them. Because of this intended general audience, we will sometimes describe, in some detail, things which are clear to a fuzzy specialist – but we would advise this specialist not to skip these parts: the things may be familiar, but our explanations for these things will usually be more foundational (and thus, different from the semi-empirical semi-heuristic explanations provided in most other texts).

Comment. It should be mentioned that while all the results presented in this paper are new, similar (preliminary) results appeared earlier in [9].

2 Need for Data Processing Naturally Leads to Fuzzy and Neural Techniques: Main Results

Decision making as an ultimate goal of data processing. The ultimate goal of data processing is to make an appropriate decision: how to treat a patient, how to drive a car, etc. In making a decision, we want to select an alternative which is the best for the decision maker.

Decision making means optimization. For each two alternatives, the decision maker, if needed, has to decide which one is better. Usually, there is some scale enabling the decision maker to describe his/her preferences by a number. For example, we can compare each alternative with monetary gains or losses, and thus find the amount of money which is equivalent to each alternative. Another scale is when we select a very good situation A_1 and a very bad situation A_0 and compare each alternative with lotteries $L(p)$ in which we have A_1 with probability p and A_0 with the remaining probability A_0.

Whatever scale we use, we assign a number $u(a)$ to each alternative a, so that the better alternatives correspond to larger values of $u(a)$. In these terms, we need to select an alternative a which maximizes the value of the objective function $u(a)$.

Dynamic character of decision making. Situations change. As a result, the objective function also changes with time: it has the form $u(t, a)$. Thus, the alternative which was the best at one moment is not the best at the next moment: a patient may develop an allergy to a medicine, a road situation may change, etc. So, we need to make decisions under such changing conditions. In other words, at each moment of time t, we need to select an alternative $a(t)$ that maximizes $u(t, a)$.

Smooth changes in objective function and resulting changes in the optimal decision. In most cases, small changes in a lead to small changes in $u(a)$; similarly, a small change in time also, in general, leads to small changes in $u(t, a)$. In precise terms, we can formulate this as requiring that $u(t, a)$ be a smooth (differentiable) function of its variables.

In general, when time t changes, the solution to the corresponding optimization problem $u(t,a) \to \max$ also changes smoothly with time – we can even describe how exactly it changes, by taking into account that the optimal solution satisfies the equation $\dfrac{\partial u}{\partial t}(t, a(t)) = 0$; by differentiating this equation with time, we get an explicit formula for $\dfrac{da}{dt}$.

However, sometimes, there is an abrupt change in optimal strategy: e.g., when the function has several local maxima, and the global maximum changes from one local maximum to another one.

A simple mathematical example is when we have a sum of two Gaussian functions $u(a) = C_- \cdot \exp(-(a+a_0)^2) + C_+ \cdot \exp(-(a-a_0)^2)$ corresponding to a large value a_0. For this function, when $C_- < C_+$, the maximum is attained for $a = a_0$, otherwise it is attained for $a = -a_0$. So, if the values $C_-(t)$ monotonically increases from 0 to 1, while the value $C_+(t)$ monotonically decreases from 1 to 0, the maximizing value a, the maximizing value x will, at some moment of time, switch from $a = -a_0$ to $a = a_0$.

The fact that optimization leads to discontinuity is well known in optimal control theory, where such a behavior is known as a "bang-bang" control; see, e.g., [4, 7].

Conclusion about data processing. On the local-time level, the optimal alternative is a smooth function of time and of other parameters describing the situation. On the global-time scale, we need to also take into account need for discrete transitions.

Comment. This conclusion is in good accordance with the fact that, e.g., when we walk, on the local-time level, we rely on an automatic neural-based mechanism for moving our legs, but on the global level, we use (discrete) reasoning to decide which way to go.

Often, decisions needs to be made fast. In many practical situations, be it a patient in crisis or a car near an unexpected obstacle, we need to make decisions in real time. It is therefore desirable to come up with algorithms that will process data as fast as possible. Let us describe, both for smooth and for discreet transformations, how this requirement translates into computations.

Case of smooth data processing: enter neural networks. Let us start with describing the fastest way to perform smooth data processing.

In general, to minimize the time of a task, be it the task of digging a canal or the task of performing computations, we need to select fastest devices for performing this task, and to use as many such devices working in parallel as possible – so that each device will get the smallest portion of the overall task and thus, finish as soon as possible. For computational tasks, this means that we need to divide the computational task into simplest (hence fastest) subtasks that will be implemented in parallel.

Which are the simplest tasks? To perform each elementary computational task, we can use different analog devices. Each such device transforms inputs

$x_1, \ldots, x_n$ into the corresponding output $y = f(x_1, \ldots, x_n)$; a natural way to use such a device for computations is to generate signals proportional to given inputs and use the resulting output. Generating such a signal takes time and effort; so, to speed up computations – and to save energy needed for running many such devices in parallel – it is desirable to use signals which are as weak as possible. In other words, we use inputs of the type $x_i = x_i^{(0)} + \Delta x_i$, where $x_i^{(0)}$ are the original values, and the values Δx_i are small. When the signals are small, any smooth function can be well approximated by its linear terms (first order terms in its Taylor expansion). Thus, the simplest possible computational tasks correspond to *linear* transformation. For electric signals, linear transformations are straightforward: e.g., the sum corresponds to a simple merging of two or more signals.

In general, the results of one computational unit serves as the input for other computational units. In mathematical terms, this means that the overall function computed by the corresponding parallel computational device is a composition of functions computed by individual units. The composition of linear functions is always linear. Thus, to perform a generic (in particular, non-linear) data processing, we need to supplement linear computational units with non-linear ones. In general, the more inputs, the longer the computations. Thus, out of all possible non-linear units, the fastest are the ones that compute functions of a single variables. Thus, we conclude that the simplest computational devices should use two types of computational units:

- linear (L) units, and
- non-linear (NL) units that perform a nonlinear transformation of one input.

A parallel computational device should thus consist of several layers of devices corresponding to different time of computations: layers of L units and layers of NL units. The fewer layers, the faster the resulting computation. Also, the fewer NL layers, the better – since L layers are the fastest. Thus, we are looking for the fastest possible combination of layers that is a *universal approximator*, i.e., that would allow us to approximate any given (continuous) function with any given accuracy. As the following result shows, this leads to usual neural networks.

Let us recall that a class of functions $\mathcal{F}$ is called a *universal approximator* if for every $\Delta > 0$ and $\varepsilon > 0$ and for every continuous functions $f(x_1, \ldots, x_n)$, there exists a function $a \in \mathcal{F}$ for which $|f(x_1, \ldots, x_n) - a(x_1, \ldots, x_n| \le \varepsilon$ for all tuples $(x_1, \ldots, x_n)$ for which $|x_i| \le \Delta$ for all i.

Proposition 1.

- *A 2-layer network is not a universal approximator.*
- *Out of possible 3-layer networks with a single NL layer, only a L-NL-L network is a universal approximator.*

Comment. Functions computed by a L-NL-L network have the following form.

- The first L layer transforms the inputs $x_1, \ldots, x_n$ into several linear combinations $\ell_k = \sum_{i=1}^{n} w_{ki} \cdot x_i + w_{k0}$.
- Then, elements forming the NL layer transform these linear combinations ℓ_k into values $y_k = s_k(\ell_k)$.
- Finally, the last L layer transforms the value y_k into their linear combination $y = \sum_k W_k \cdot y_k + W_0$.

The resulting dependence

$$y = \sum_k W_k \cdot s_k \left(\sum_{i=1}^{n} w_{ki} \cdot x_i + w_{k0} \right) + W_0$$

is exactly what is computed by a 3-layer neural network; see, e.g., [1]. Thus, *smooth data processing naturally leads to neural networks.*

Case of discrete (non-analog) data processing. For discrete data processing, the fastest processing of two data values x_1 and x_2 is when we do not perform any data processing at all, i.e., when we always return either x_1 or x_2. A simple result shows that there are only two non-trivial functions with this property.

Proposition 2. *If $f(x_1, x_2)$ is a continuous function whose is always equal to either x_1 or x_2 and which is not always equal to x_1 and not always equal to x_2, then either $f(x_1, x_2) = \min(x_1, x_2)$ or $f(x_1, x_2) = \max(x_1, x_2)$.*

One might argue that the actual comparison also takes some computation time. This is true, but, as the following result shows, this time is miniscule:

Proposition 3. *For every n, deciding which of two n-bit numbers x_1 and x_2 is smaller requires, on average, ≤ 2 bit operations.*

So, in the discrete case, the fastest operations are min and max. Since both functions simply select one of the inputs, any composition of these functions will also only select one of the inputs. So, to represent functions whose values differ from all the inputs, we need to supplement such operations with non-linear functions. As as have mentioned earlier, the simplest non-linear functions are functions of one variable. Thus, here we have layers of three types: Min, Max, and NL (functions of one variable).

Proposition 4.

- *2-layer networks are not universal approximators.*
- *Out of possible 3-layer networks with a single NL layer, only NL-Min-Max and NL-Max-Min networks are universal approximators.*

Functions computed by NL-Min-Max networks have the form $f(x_1, \ldots, x_n) = \max(R_1(x_1, \ldots, x_n))$, where $R_k(x_1, \ldots, x_n) = \min(f_{k1}(x_1), \ldots, f_{kn}(x_n))$. These functions are very common in fuzzy data processing: they comes from *Mamdani approach*, where degrees $R_k(x_1, \ldots, x_n)$ corresponding to different rules are combined by an "or"-operation (in this case, max), and the membership functions $f_{ki}(x_i)$ corresponding to each rule are combined by an "and"-operation (in this case, min). To be more precise, in general, in the Mamdani's approach, the rule base is applicable if one of the rules is applied, and the applicability of an implication rule $A_{k1} \,\&\, \ldots \,\&\, A_{k,n-1} \to A_{kn}$ is interpreted as the fact that its conditions are satisfied and its conclusion is satisfied, i.e., as a conjunction $A_{k1} \,\&\, \ldots \,\&\, \neg A_{k,n-1} \,\&\, A_{kn}$.

Functions computed by NL-Max-Min networks have the form $f(x_1, \ldots, x_n) = \min(R_1(x_1, \ldots, x_n))$, where $R_k(x_1, \ldots, x_n) = \max(f_{k1}(x_1), \ldots, f_{kn}(x_n))$. These functions correspond to the *logical approach* to fuzzy modeling and fuzzy control, where degrees $R_k(x_1, \ldots, x_n)$ corresponding to different rules are combined by an "and"-operation (in this case, min), and the membership functions $f_{ki}(x_i)$ corresponding to each rule are combined by an "or"-operation (in this case, max). To be more precise, in general, in the logical approach, the rule base is applicable if all the rules are applicable, and the implication $A_{k1} \,\&\, \ldots \,\&\, A_{k,n-1} \to A_{kn}$ forming each rule is interpreted as an equivalent disjunction $\neg A_{k1} \vee \ldots \vee A_{k,n-1} \vee A_{kn}$. Thus, *discrete data processing naturally leads to fuzzy techniques.*

Mamdani approach or logical approach? We have just shown that there are two ways to approximate a general function by fuzzy data processing: as an NL-Min-Max network corresponding to Mamdani approach, and as an NL-Max-Min network corresponding to the logical approach. Which of these two approaches is preferable?

From the pure computational viewpoint of approximation ability, both approaches work well. However, one of the important advantages of fuzzy approach in general is that the corresponding rules are *interpretable*. From this viewpoint, as we will see, these two approaches differ. As we can see from the proof of Proposition 4, when all the values $f(x_1, \ldots, x_n)$ of the approximated function are in the interval $[0, 1]$, then with an NL-Max-Min network, we get an approximation in which all corresponding membership functions $1 - f_i^{(\ell)}(x_i)$ are *normalized* – i.e., the maximum of each of these functions is equal to 1. On the other hand, the approximation that we provided for a Mamdani-type NL-Min-Max network has a non-normalized function, with $\max\limits_{x_1}(f_1^{(\ell)}(x_1)) = f(x^{(\ell)}) < 1$ for some ℓ. It turns out that this is not just a limitation of our proof, it is a general limitation of Mamdani-type NL-Min-Max networks:

Proposition 5.

- *Mamdani-type NL-Min-Max networks with normalized membership functions* are not *universal approximators for continuous functions* $f(x) \in [0, 1]$.

- *Logical-type NL-Max-Min networks with normalized membership functions* are *universal approximators for continuous functions* $f(x) \in [0,1]$.

What if we have limited parallelism. The above justifications apply when we have unlimited parallelism. When the parallelization abilities are limited, we may need more layers to perform data processing. Some example of such multi-layer generalizations are actually efficiently used.

Example 1: a generalization of 3-layer neural data processing. Another example is *deep neural networks*, where more than 3 layers are used; see, e.g., [2].

Example 2: a generalization of min-max fuzzy data processing. What if, in Mamdani's approach, we use general t-norms and t-conorms instead of min and max?

A general t-norm can be approximated, with any given accuracy, by an Archimedean t-norm which has the form $f_{\&}(a,b) = \psi(\psi^{-1}(a) + \psi^{-1}(b))$; see, e.g., [10]. Similarly, a general t-conorm can be approximated by an Archimedean t-conorm of the type $f_{\vee}(a,b) = \varphi(\varphi^{-1}(a) + \varphi^{-1}(b))$. The resulting Mamdani-type function $f(x_1,\ldots,x_n) = f_{\vee}(R_1(x_1,\ldots,x_n),\ldots,R_k(x_1,\ldots,x_n),\ldots)$, where $R_k(x_1,\ldots,x_n) = f_{\&}(\mu_{k1}(x_1),\ldots,\mu_{kn}(x_n))$, can be computed by a network of the type NL-L-NL-L-NL.

Example 3: a generalization of both neural and fuzzy data processing techniques. What if we combine fast layers corresponding to continuous and discrete data processing, i.e., linear, Min, and Max layers?

One possibility is to have Min-Max-L or Max-Min-L networks. They correspond to a linear combination of different elements in the ordering $x_{(1)} < \ldots < x_{(n)}$ of the original values $x_1,\ldots,x_n$, a particular case of which is Yager's Ordered Weighted Average (OWA) combinations; see, e.g., [14].

These schemes are not universal approximators, since these functions are linear on n subdomains.

Comment. It is worth mentioning that similar schemes L-Max-L and L-Min-L are universal approximators. Indeed, each continuous function can be represented as the difference of two convex functions, and each convex function can be represented as a maximum of linear functions; see, e.g., [13].

3 What Is the Meaning of Fuzzy Degrees?

What is the meaning of fuzzy degrees: formulation of the problem. The above results show that fuzzy techniques *can* be, in principle, effectively used in data processing. A natural question is: *how* can we use these techniques? how can we elicit the corresponding fuzzy degrees? In other words, what is the operational meaning of fuzzy techniques?

People can usually easily qualitatively *compare* their degree of confidence in different statements, but what is needed is a *quantitative* description of these

degrees of certainty. Some people have no problem marking their degrees on a numerical scale, but for others, this is a big problem, since to them, it is not clear what exactly is the meaning of estimates like "8 on a scale from 0 to 10". We thus need to come up with an operational meaning of fuzzy techniques.

Analysis of the problem. Fuzzy degrees themselves are not directly observable. What *is* observable is the result $\overline{x}$ of applying a defuzzification procedure to the given membership degrees $\mu(x)$. Let us use this fact to explain how fuzzy degrees can get an operational meaning.

Resulting meaning of fuzzy degrees: case when we fix a defuzzification procedure. Let us start with the case when the defuzzification procedure is fixed – e.g., as the centroid defuzzification. In this case, when the expert expresses his or her degree of confidence by a certain word (or combination of words) w from natural language, a reasonable way to elicit the corresponding degree $\mu \in [0,1]$ is to ask, to this expert, a question of the following type:

- assume that we have two possible values x_0 and x_1, the value x_0 is definitely possible, while the value x_1 is possible with degree w;
- what is then a reasonable control value $\overline{x}$ to select?

For example, if we use centroid defuzzification, then a reasonable choice for $\overline{x}$ will be $\overline{x} = \dfrac{x_0 + \mu \cdot x_1}{1+\mu}$. Thus, once we know the appropriate value $\overline{x}$, we can determine the corresponding degree μ as the value for which this equality is true, i.e., as $\mu = \dfrac{\overline{x} - x_0}{x_1 - \overline{x}}$.

Comment. In some cases, fuzzy degrees have a probabilistic meaning: e.g., if we have several experts, we can estimate the degree to which a certain value x is small, by taking the ratio of how many experts consider the value x to be small. The above interpretation provides us with a non-probabilistic operational meaning of fuzzy degrees – in perfect accordance with Zadeh's idea (see, e.g., [16]) that fuzzy information cannot be always deescribed in probabilistic terms.

General case: we need different defuzzification procedures. The above procedure works OK if an expert is unable to assign the degree him/herself. But what is an expert is quite capable of assigning such degrees?

It is known that different experts sometimes assign different numerical values to the same degree of certainty; some consistently assign lower values, some higher values. This is similar to grading: even when instructors fully agree on whose test results are better and whose are worse, some instructors consistently assign higher numerical values to all the students.

In this case, two experts assign different degrees $\mu(x) < \mu'(x)$, but their expected value $\overline{x}$ should be the same. To come up with the same value $\overline{x}$ corresponding to two different functions $\mu(x) \neq \mu'(x)$, we need, in general, to use *different defuzzification procedures*. In other words, we need to *adjust* a defuzzification procedure to an individual expert.

Let us give an example. Suppose that the opinions of expert A are well described by centroid defuzzification: once we know his degrees of confidence $\mu(x)$, we can estimate this expert's recommended value as $\overline{x} = \dfrac{\int x \cdot \mu(x)\,dx}{\int \mu(x)\,dx}$. Suppose then that expert B routinely assigns larger values $\mu'(x) = \sqrt{\mu(x)}$ to describe the same degrees of confidence. In this case, to come up with the same recommended value $\overline{x}$, we need to first transform B's degrees into A's scale, i.e., compute the values $\mu(x) = (\mu'(x))^2$, and then apply the centroid defuzzification. This is equivalent to applying a new defuzzification $\overline{x} = \dfrac{\int x \cdot (\mu'(x))^2\,dx}{\int (\mu'(x))^2\,dx}$ to B's membership function $\mu'(x)$.

In general, if $\mu(x) = F(\mu'(x))$, for some monotonic function $F(x)$, then, instead of the centroid defuzzification, we should use a new procedure $\overline{x} = \dfrac{\int x \cdot F(\mu'(x))\,dx}{\int F(\mu'(x))\,dx}$.

Conclusion: fuzzy degrees themselves do not have exact meaning. Our conclusion is that fuzzy degrees, by themselves, do not have any exact meaning, they gain meaning only we combine them with a defuzzification procedure. In other words, fuzzy degrees themselves do not have an operational meaning; what has an operational meaning is a combination of fuzzy degrees *and* a defuzzification procedure – and if we re-scale fuzzy degrees, we get the same observable results if we corresponding change the defuzzification procedure.

Comment. In view of this year's 100th anniversary of Einstein's General Relativity, it is worth mentioning that in General Relativity, as the famous mathematician H. Poincaré noticed in the early 20 century [3, 8], there is a similar phenomenon: we cannot directly observe geometry or physics, we can only observe geometry + physics, so that a change in geometry can be compensated by the appropriate change in physical equations.

For example, we can stick with the curved-space geometry of General Relativity theory, and conclude that the free particles follow geodesics – shortest lines in this curved space. Alternatively, we can assume that the space is Euclidean, but there is an additional gravitational force that curves the trajectories of all free-moving particles.

4 Auxiliary Result: What Are Reasonable Defuzzification Operations

Formulation of the problem. In the previous section, we showed that to explain an operational meaning of fuzzy techniques, we must take into account defuzzification operations. It is therefore reasonable to ask: what are reasonable defuzzification operations?

Main idea. Our main idea is that a defuzzification operation, when applied to values $x_1, \ldots, x_n$, should return a single value $\overline{x}$ *and* it should also return a

"degree" μ explaining how must trust in this value $\overline{x}$. These two values should contain most original information about the values and their degrees – so that when we get an additional value x_{n+1} with a new degree μ_{n+1}, we should get the same result, whether we combine all $n+1$ pairs (x_i, μ_i), or whether we combine the new pair (x_{n+1}, μ_{n+1}) with the combined pair $(\overline{x}, \mu)$.

Additional idea. In many cases, the values x_i represent numerical values of a physical quantity such as temperature, time, distance, etc. The numerical value of a physical quantity depends on which starting point we use to measure this quantity, and which measuring unit we use. For example, the Celsius and Fahrenheit temperature scales differ both by starting points and by units. If we change a starting point by x_0, the numerical value changes from x to $x' = x + x_0$. If we replace the original measuring unit by a new unit which is λ times smaller, we get a new numerical value $x' = \lambda \cdot x$. If we change both the starting point and the measuring unit, then the new numerical values can be obtained from the previous ones by a linear transformation $x' = \lambda \cdot x + s_0$.

These different numerical values x and x' correspond to the same actual value of the physical quantity: same temperature, same speed, etc. It is therefore reasonable to require that if we apply the defuzzification procedure to new numerical values $x'_1, \ldots, x'_n$, then we should get the same result as before, but expressed in the new scale.

Definition. *By a* reasonable defuzzification*, we mean a mapping that maps each finite collection of pairs (x_i, μ_i) with different values x_i into a single pair $(\overline{x}, \mu)$ so that the following two properties are satisfied:*

- *for every sequence (x_1, μ_1), …, (x_n, μ_n), (x_{n+1}, μ_{n+1}), the result of combining these $n+1$ pairs coincides with the result of combining the two pairs $(\overline{x}((x_1, \mu_1), \ldots, (x_n, \mu_n)), \mu((x_1, \mu_1), \ldots, (x_n, \mu_n)))$ and (x_{n+1}, μ_{n+1});*
- *for every sequence (x_1, μ_1), …, (x_n, μ_n), and for every real numbers $\lambda \neq 0$ and x_0, the value $\overline{x}$ obtained by combining these pairs and the value $\overline{x'}$ obtained by combining pairs $(\lambda \cdot x_i + x_0, \mu_i)$ are related by the formula $\overline{x'} = \lambda \cdot \overline{x} + x_0$.*

Proposition 6. *Every reasonable defuzzification has the form* $\overline{x} = \dfrac{\sum\limits_{i=1}^{n} \varphi(\mu_i) \cdot x_i}{\sum\limits_{i=1}^{n} \varphi(\mu_i)}$ *for some function $\varphi(x)$. Vice versa, every operation of this type is a reasonable defuzzification – when combined with an appropriate function*

$$\mu((x_1, \mu_1), \ldots, (x_n, \mu_n)).$$

Discussion. Thus, in effect, the only reasonable defuzzification is a centroid defuzzification – after an appropriate re-scaling of fuzzy values, from the original values μ_i to re-scaled values $\varphi(\mu_i)$.

Acknowledgments

This work was supported in part by the Faculty of Economics, Chiang Mai University, and by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721.

References

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*, Springer Verlag, New York, 2013.

[2] L. Deng and D. Yu, *Deep Learning: Methods and Applications*, Now Publ., Boston, Massachusetts, 2014.

[3] R. P. Feynman, R. B. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison-Wesley, Boston, Massachusetts, 2005.

[4] D. E. Kirk, *Optimal Control Theory: An Introduction*, Dover, New York, 2004.

[5] G. J. Klir and B. Yuan, *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, Prentice Hall, Upper Saddle River, New Jersey, 1995.

[6] V. Kreinovich, M. C. Mouzouris, and H. T. Nguyen, "Fuzzy rule based modeling as a universal approximation tool", In: H. T. Nguyen and M. Sugeno (eds.), *Fuzzy Systems*, 1998, p. 135–195.

[7] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal Control*, Wiley, Hoboken, New Jersey, 2012.

[8] C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation*, W. H. Freeman, San Francisco, California, 1973.

[9] H. T. Nguyen, V. Kreinovich, F. Modave, and M. Ceberio, "Fuzzy without fuzzy: why fuzzy-related aggregation techniques are often better even in situations without true fuzziness", In: A. E. Hassanien et al. (eds.), *Foundations of Computational Intelligence*, Vol. 2, Springer Verlag, 2009, pp. 27–51.

[10] H. T. Nguyen, V. Kreinovich, and P. Wojciechowski, "Strict archimedean t-norms and t-conorms as universal approximators", *International Journal of Approximate Reasoning*, 1998, Vol. 18, No. 3-4, pp. 239–249.

[11] H. T. Nguyen and E. A. Walker, *A First Course in Fuzzy Logic*, Chapman and Hall/CRC, Boca Raton, Florida, 2006.

[12] I. Perfilieva and V. Kreinovich, "A new universal approximation result for fuzzy systems which reflects CNF-DNF duality", *International Journal of Intelligent Systems*, 2002, Vol. 12, pp. 1121–1130.

[13] R. T. Rockafeller, *Convex Analysis*, Princeton University Press, Princeton, New Jersey, 1997.

[14] R. R. Yager and J. Kacprzyk (eds.), *The Ordered Weighted Averaging Operators: Theory and Applications*, Springer Verlag, Berlin, Heidelberg, New York, 2012.

[15] L. A. Zadeh, "Fuzzy sets", *Information and Control*, 1965, Vol. 8, pp. 338–353.

[16] L. A. Zadeh, "The information principle", *Abstracts of the International IEEE Conference on Systems, Man, and Cybernetics IEEE SMC'2014*, San Diego, Florida, October 5–8, 2014.

A Proofs

Proof of Proposition 1. Composition of two linear functions is linear, composition of two functions of one variable is also a function of one variable. Thus, we can always collapse two neighboring L layers or two neighboring NL layers into one. Hence, it is sufficient to only consider configurations in which L and NL layers interchange.

For 2 layers, this means L-NL or NL-L. In the first case, we only have functions of the type $f(x_1, \ldots, x_n) = s\left(\sum\limits_i w_i \cdot x_i + w_0\right)$. For these functions, the level sets $\{x : f(x) = c\}$ are unions of subspaces of dimension $n-1$. Thus, they cannot approximate, e.g., multiplication $f(x_1, x_2) = x_1 \cdot x_2$ for which the level sets are hyperbolas.

In the second case, we only have functions of the type

$$f(x_1, x_2) = \sum_{i=1}^{n} w_i \cdot s_i(x_i) + w_0.$$

For such functions, for all possible pairs (x_1, x_2) and (x_1', x_2'), we have $f(x_1, x_2) + f(x_1', x_2') = f(x_1, x_2') + f(x_1', x_2)$. Clearly, the multiplication function for which $1 = f(0,0) + f(1,1) \neq f(0,1) + f(1,0) = 0$, cannot be approximated by such functions.

For 3 layers with one NL layer, the only option is L-NL-L, and it is known – from the theory of neural networks – that such networks are indeed universal approximators; see, e.g., [1].

Proof of Proposition 2. On the half-plane $\{(x_1, x_2) : x_1 < x_2\}$, every two points are connected by a continuous line interval which is strictly within this half-plane. Since the function $f(x_1, x_2)$ is continuous, it cannot switch from being equal to x_1 at one point to being x_2 at another point, since then at the connecting interval, we will have a discontinuity. Thus, either for all these points, we have $f(x_1, x_2) = x_1$, or for all of them, we have $f(x_1, x_2) = x_2$. Due to continuity, the same holds in the limit case $x_1 = x_2$.

Similarly, either for all points (x_1, x_2) for which $x_1 \geq x_2$, we have $f(x_1, x_2) = x_1$ or for all of them, we have $f(x_1, x_2) = x_2$. We cannot have $f(x_1, x_2) = x_1$ for both cases, so if $f(x_1, x_2) = x_1$ when $x_1 \leq x_2$, then we should have $f(x_1, x_2) = x_2$ when $x_1 \leq x_2$. In this case, we have $f(x_1, x_2) = \min(x_1, x_2)$. In the second case, we have $f(x_1, x_2) = \max(x_1, x_2)$. The proposition is proven.

Proof of Proposition 3. With probability 1/2, the first bits of the numbers x_1 and x_2 are different, i.e., one of these bits is 0 and another one is 1. In this case, after a single but comparison, we know which number is smaller: the one that starts with 0.

In the remaining cases, in half of these cases, the second bits are different, and so we get a solution after 2 bit operations. The probability of this situation is 2^{-2}. In general, for every k, we need k bit comparisons with probability 2^{-k}. Thus, the average computation time is equal to $\sum\limits_{k=1}^{n} 2^{-k} \cdot k$. This value is bounded from above by the corresponding infinite sum $s \stackrel{\text{def}}{=} \sum\limits_{k=1}^{\infty} 2^{-k} \cdot k$.

One can easily check that

$$2^{-1} \cdot s = \sum_{k=1}^{\infty} 2^{-(k+1)} \cdot k = \sum_{k=1}^{\infty} 2^{-(k+1)} \cdot (k+1) - \sum_{k=1}^{\infty} 2^{-(k+1)}.$$

The first term in this sum is $s - 2^{-1} \cdot 1 = s - 2^{-1}$, the second is a geometric progression whose sum is 2^{-1}. Thus, $2^{-1} \cdot s = s - 2^{-1} - 2^{-1}$, hence $s = 2$. The proposition is proven.

Proof of Proposition 4. Similarly to the proof of Proposition 1, we can collapse similar neighboring layers into one. Thus, it is sufficient to consider networks in which neighboring layers are different.

For 2 layers, this means Min-NL, Max-NL, NL-Min, and NL-Max (we already know that Min-Max and Max-Min are not universal approximators). A Min-NL function $f(x_1, x_2) = s(\min(x_1, x_2))$ depends only on x_1 when $x_1 < x_2$ and thus, cannot approximate $x_1 \cdot x_2$. Similarly, Max-NL functions are not universal approximators.

An NL-Min function has the form $f(x_1, x_2) = \min(s_1(x_1), s_2(x_2))$. For such functions, $\min(f(x_1, x_2), f(x_1', x_2')) = \min(f(x_1, x_2'), f(x_1', x_2))$. Thus, e.g., the function $f(x_1, x_2) = 1 - x_1 \cdot x_2$ for which $0 = \min(f(0,0), f(1,1)) \neq \min(f(0,1), f(1,0)) = 1$ cannot be thus approximated. Similarly, NL-Max networks are not universal approximators.

A Max-Min-NL or a Min-Max-NL function has the form $s(x_i)$ for one of the values x_i, i.e., its domain can be divided into n domains on each of which it is equal to a function of one of the inputs – this cannot approximate multiplication. A Max-NL-Max or Min-NL-Max function locally has the form $\max\limits_i(s_i(x_i))$, and we have already shown that such functions are not universal approximators. Similarly, Max-NL-Min and Min-NL-Min functions are not universal approximators.

For functions NL-Max-Min and NL-Min-Max, universal approximation property is known; see, e.g., [5, 6, 11, 12]. Since our formulation is slightly different from the usual one, let us provide the proof explicitly; this proof is very similar to the usual proofs.

Let us start with NL-Min-Max networks. Let $f(x_1, \ldots, x_n)$ be a continuous function, and let $\Delta > 0$ and $\varepsilon > 0$ be real numbers. Since the function $f(x_1, \ldots, x_n)$ is continuous on a compact cube $[-\Delta, \Delta] \times \ldots \times [-\Delta, \Delta]$, it is uniformly continuous, and thus, there exists a real number $\delta > 0$ for which if $|x_i - x_i'| \le \delta$ for all i, then $|f(x_1, \ldots, x_n) - f(x_1', \ldots, x_n')| \le \varepsilon$. On each interval $[-\Delta, \Delta]$, let us select a grid of values with a step $\delta/2$, i.e., the values $x_i = -\Delta$, $x_i = -\Delta + \delta/2$, $x_i = -\Delta + \delta$, ... We then take all possible tuples $x^{(\ell)} = (x_1^{(\ell}, \ldots, x_n^{\ell)})$ formed by these values. Let us then show that the NL-Min-Max function

$$a(x_1, \ldots, x_n) = \max_{\ell} \min_{i=1,\ldots,n} f_i^{(\ell)}(x_i)$$

is the desired ε-approximation to the original function $f(x_1, \ldots, x_n)$, where $f_1^{(\ell)}(x_1) = f(x^{(\ell)}) \cdot \mu_0(x_1 - x_1^{(\ell)})$, $f_i^{(\ell)}(x_i) = \mu_0(x_i - x_i^{(\ell)})$ for all $i \ge 2$, and by $\mu_0(x)$, we denoted a piece-wise linear function which is equal to 0 when $|x| \ge \delta$, to 1 when $|x| \le \delta/2$, and which is linear on the intervals $[-\delta, -\delta/2]$ and $[\delta/2, \delta]$.

Indeed, the only tuple ℓ for which $\min\limits_i \mu_0(x_i - x_i^{(\ell)}) > 0$, we have $\mu_0(x_i - x_i^{(\ell)}) > 0$ for all i. Thus, by definition of the function $\mu_0(x)$, we have $|x_i - x_i^{(\ell)}| \le \delta$ for all i. By the choice of δ, this implies that $|f(x) - f(x^{(\ell)})| \le \varepsilon$ and thus, $f(x^{(\ell)}) \le f(x) + \varepsilon$. The product of functions $\mu_0(x)$ is always smaller than or equal to 1, thus, all the terms $f(x^{(\ell)}) \cdot \min\limits_i \mu_0(x_i - x_i^{(\ell)})$ do not exceed $f(x) + \varepsilon$. Therefore, the value $a(x)$ which is the largest of these values also does not exceed $f(x) + \varepsilon$: $a(x) \le f(x) + \varepsilon$.

On the other hand, because of our choice of the points $x^{(\ell)}$, for each point x, there is a point $x^{(\ell)}$ for which all coordinates are $(\delta/2)$-close to the coordinated of the original point x. In this case, each of the functions $\mu_0(x_i - x_i^{(\ell)})$ is equal to 1, their product is equal to 1, and thus, the corresponding term

$$f(x^{(\ell)}) \cdot \min_i \mu_0(x_i - x_i^{(\ell)})$$

is equal to $f(x^{(\ell)}$ and is, hence, greater than or equal to $f(x) - \varepsilon$. The maximum $a(x)$ is larger than or equal than each of the maximized terms, so $a(x) \ge f(x) - \varepsilon$. With $a(x) \le f(x) + \varepsilon$, this means that $|a(x) - f(x)| \le \varepsilon$.

The possibility to approximate a function $f(x)$ by a NL-Max-Min expressions comes from the fact that $g(x) = 1 - f(x)$ can be approximated by a NL-Min-Max function $a(x)$, a function of the type $a(x) = \max\limits_{\ell} \min\limits_i f_i^{(\ell)}(x_i)$. Since the functions $a(x)$ and $1 - f(x)$ are ε-close, we can conclude that the functions $f(x)$ and $b(x) \stackrel{\text{def}}{=} 1 - a(x)$ are also ε-close. One can see that the function $b(x) = 1 - a(x)$ can be described in NL-Max-Min terms as $b(x) = \min\limits_{\ell} \max\limits_i (1 - f_i^{(\ell)}(x_i))$. The proposition is proven.

Proof of Proposition 5. In Proposition 4, we have, in effect, already proved that NL-Max-Min networks with normalized membership functions are universal approximators for continuous functions $f(x)$ with values from the interval $[0,1]$. So, to prove Proposition 5, it is sufficient to prove that NL-Min-Max networks with normalized functions $f_i^{(\ell)}(x_i)$ are *not* universal approximators for functions $f(x) \in [0,1]$.

Indeed, let us show that, for example, a function $f(x) = 0.5$ cannot be thus approximated. Infeed, any such approximation has the form $a(x_1,\ldots,x_n) = \max\limits_{\ell} \min\limits_{i} f_i^{(\ell)}(x_i)$ for some functions $f_i^{(\ell)}$ which are normalized. Normalization means that for each such function, there is a value $x_i^{(\ell)}$ for which $f_i^{(\ell)}(x_i^{(\ell)}) = 1$. For the tuple $x^{(\ell)} \stackrel{\rm def}{=} (x_1^{(\ell)},\ldots,x_n^{(\ell)})$, we thus have $\min\limits_{i} f_i^{(\ell)}(x_i^{(\ell)}) = 1$, thence $a(x)$, which is the largest of such values, is also equal to 1 – and therefore, cannot approximate a function $f(x)$ whose value everywhere is 0.5. The proposition is proven.

Proof of Proposition 6. Let us first consider the case when we combine two pairs (x_1,μ_1) and (x_2,μ_2). By an appropriate linear transformation, we can transform 0 to x_1 and 1 to x_2: namely, this transformation has the form $x' = \lambda\cdot x + x_0$, where $x_0 = x_1$, and $\lambda = x_2 - x_1$. Thus, due to the second property of a reasonable defuzzification, the result $(\overline{x},\mu)$ of combining the two original pairs is equal to $(\lambda\cdot\overline{x}_0 + s_0,\mu)$, where $\overline{x}_0 = \overline{x}((0,\mu_1),(1,\mu_2))$. Hence,

$$\overline{x} = (x_2 - x_1)\cdot\overline{x}((0,\mu_1),(1,\mu_2)) + x_1 =$$

$$x_1\cdot(1-\overline{x}((0,\mu_1),(1,\mu_2))) + x_2\cdot\overline{x}((0,\mu_1),(1,\mu_2)).$$

In other words, the result of combining the two pairs (x_1,μ_1) and (x_2,μ_2) is a linear combination of values x_1 and x_2, with coefficients depending only on the degrees μ_i.

Due to the first property from the definition of a reasonable defuzzification, the result of combining n pairs can be obtained by first combining the first two, then by adding the third pair, etc. Thus, we can conclude that this combination result is a linear combination of the values x_i, where the weights depend only on the degrees $\mu_1,\ldots,\mu_n$:

$$\overline{x} = \sum_{i=1}^{n} w_i(\mu_1,\ldots,\mu_n)\cdot x_i.$$

The need to have scale-invariance (i.e., invariance with respect to re-scaling $x \to x' = \lambda\cdot x$) leads to $\sum\limits_{i=1}^{n} w_i = 1$.

When we combine the first pair, we get a linear combination

$$w_1(\mu_1,\mu_2)\cdot x_1 + w_2(\mu_1,\mu_2)\cdot x_2.$$

On all following steps, we use this whole combination to combine with other values x_3, etc. Thus, the ratio of the coefficients at w_1 and w_2 remains the

same. Thus, this ratio depends only on μ_1 and μ_2:

$$\frac{w_1(\mu_1, \mu_2, \ldots, \mu_n)}{w_2(\mu_1, \mu_2, \ldots, \mu_n)} = r(\mu_1, \mu_2),$$

where $r(\mu_1, \mu_2) \stackrel{\text{def}}{=} \frac{w_1(\mu_1, \mu_2)}{w_2(\mu_1, \mu_2)}$. In general, $\frac{w_1}{w_2} = \frac{w_1}{w_3} : \frac{w_2}{w_3}$. Thus, $r(\mu_1, \mu_2) = \frac{r(\mu_1, \mu_3)}{r(\mu_2, \mu_3)}$. This is true for all possible values μ_3, in particular, for $\mu_3 = 1$. Hence, $r(\mu_1, \mu_2) = \frac{\varphi(\mu_1)}{\varphi(\mu_2)}$, where we denoted $\varphi(z) \stackrel{\text{def}}{=} r(z, 1)$. From $r(\mu_1, \mu_2) = \frac{w_1}{w_2} = \frac{\varphi(\mu_1)}{\varphi(\mu_2)}$, we conclude that $\frac{w_1}{\varphi(\mu_1)} = \frac{w_2}{\varphi(\mu_2)}$. This is true for any two pairs. Thus, the ratio $c \stackrel{\text{def}}{=} \frac{w_i}{\varphi(\mu_i)}$ does not depend on i, and $w_i = c \cdot \varphi(\mu_i)$. From the condition that $\sum_{i=1}^{n} w_i = 1$, we can find $c = \frac{1}{\sum_{i=1}^{n} \varphi(\mu_i)}$. The proposition is proven.