Why Do We Need . . . ?
(in alphabetic order)

. . . α-cuts? In some situations – e.g., when we have an obstacle in front a car and we can swerve:

- either to the left
- or to the right,

centroid defuzzification leads to hitting this obstacle. To avoid such disastrous recommendation, we need to limit the set of possible controls to values which are sufficiently reasonable – i.e., for which the degree of reasonableness is not smaller than a certain threshold value α (pronounced *alpha*). This set is exactly the α-cut.

. . . activation functions? In neural networks, data processing consists of interchangingly applying linear and nonlinear transformations. If we only had linear transformations, we would be able to only compute linear functions, and many real-life dependencies are nonlinear. So, we need nonlinear transformations, and such transformations are exactly what is called activation functions.

. . . "and"-operations? By asking experts and interpolating, we can get degrees of confidence in statements like "the difference in temperatures ΔT is small positive" and "control u is small negative". However, to describe to what extent the rule is applicable, we need to know the degree of confidence in an "and"-combination: e.g.,

$$\Delta T \text{ is small positive } \textit{and } u \text{ is small negative.}$$

We need to estimate this degree based on the known degrees of confidence in statements "ΔT is small positive" and "u is small negative". To perform such estimations, we need "and"-operations.

. . . defuzzification? By applying fuzzy techniques to the expert rules, for each possible value u of control, we can generate the degree $\mu(u)$ to which this value is reasonable. As a result, we get what is called a membership function (or a fuzzy set) $\mu(u)$.

If we are designing an automatic system, then we need to generate a *single* control value $\overline{u}$ that the system will apply. So, we need to transform the fuzzy set into an exact value. This transformation is known as *defuzzification.*

. . . explainable AI? Many AI programs – in particular, the ones that use deep learning – just provide a recommendation, they do not come with any explanation. We know that these programs are not perfect, that sometimes their recommendations are wrong – but since there are no explanations, we do not know which recommendations are wrong. It is therefore desirable to have such explanations.

. . . fuzzy techniques? A large part of expert experience – namely, the rules the experts formulate in terms of imprecise words from natural language – are not used in automatic control, because computer systems do not understand natural language. Fuzzy techniques translate experts' natural-language rules into precise computer-understandable terms – i.e., into numbers.

. . . individual robustness? Our precise description of robustness means that, e.g., the values $f(a)$ and $f(a')$ corresponding to close a and a' are, *on average*, close to each other. This makes sense if we control, e.g., a flock of UAVs for studying weather. Even if one of them fails, we still have a good picture of the weather if most of all are successful and follow the desired trajectory.

Sometimes, we need individual robustness. In some situations, we are interested in the success of an individual object – e.g., we have a single UAV. In this case, the fact that most other UAVs – that, e.g., collect weather information in other cities – will be successful is no help if the UAV collecting weather information in our city of El Paso fails.

To deal with such situations, we do not just want to require that sum of the squares of the differences is small, we want to require that each difference is small, i.e., in terms of a function $f(x)$:

- if the values x and x' are close, then
- *each* corresponding pair $f(x)$ and $f(x')$ should also be close.

This is exactly individual robustness.

. . . interpolation/extrapolation in fuzzy techniques? To describe a natural-language property like "small", fuzzy technique assigns, to each value x of the corresponding quantity, the degree $\mu(x)$ to which this value satisfies the given property – e.g., to which x is small. These degrees should come from the experts. However, there are infinitely many possible values x, but we can only asked finitely many questions. Thus, from the experts, we can only get finitely many values $\mu\left(x^{(1)}\right)$, $\mu\left(x^{(2)}\right)$, . . . To get the values $\mu(x)$ for all other x, we need to use interpolation/extrapolation.

. . . max-pooling? To make sure that the pooling is performed as fast as possible.

. . . "or"-operations? By applying "and"-operations, we can find to what degree each rule is applicable. However, what we need is the degree to which the control is reasonable, i.e., to which

the first rule is applicable *or* the second rule is applicable.

We need to estimate this degree based on the known degrees of confidence that each rules is applicable. To perform such estimations, we need "or"-operations.

. . . pooling? One of the main applications of neural networks is to process pictures. In a computer, a picture is represented by storing intensity values – or, for color pictures, intensity values corresponding to three basic colors – for each pixel, and there are millions of pixels. Processing all these millions of values would take a lot of time.

To save this time, we can use the fact that for most images:

- once we know what is in a given pixel,
- we can expect approximately the same information in the neighboring pixels.

Thus, to save time, instead of processing each pixel one by one, we can combine ("pool") values from several neighboring pixels into a single value.

. . . rectified linear activation functions? To make sure that, when trained, the neural network computes the result as fast as possible.

. . . robust "and"- and "or"-operations? The expert estimate depends on a scale. If we ask the expert to estimate the degree on a scale from 0 to 5, then possible values of the resulting degree are:

$$0/5 = 0.0;\quad 1/5 = 0.2;\quad 2/5 = 0.4;\quad 3/5 = 0.6;\quad 4/5 = 0.8;\ \text{and}\ 5/5 = 1.0.$$

However, if we ask the same expert to estimate his/her degree on a scale from 0 to 4, then we will get different possible values:

$$0/4 = 0.0;\quad 1/4 = 0.25;\quad 2/4 = 0.5;\quad 3/4 = 0.75;\ \text{and}\ 4/4 = 1.0.$$

Suppose that in the first scale, the expert marked 4 on a scale from 0 to 5, leading to an estimate of 0.8. However, no mark on a 0 to 4 scale will lead to the same value 0.8; the closest is the value 0.75 which corresponds to 3 on the 0 to 4 scale. The value 0.75 is close to 0.8, but different.

Similar problem occurs if we use polling: for different numbers of experts, we get different values describing the same degrees of belief. In both cases, the same confidence level of an expert leads, in general, to different degrees $a \neq a'$ – depending on the scale or on the number of experts.

It is therefore reasonable to require that the corresponding small difference $a' - a$ should affect the results as little as possible, i.e., that the "and"- and "or"-operations be robust.

. . . robust membership functions? In practical applications, the value of the quantity x comes from measurements, and measurements are never absolutely accurate. Anyone who ever measured anything – be it voltage, current, blood pressure, whatever – knows that if we repeat the measurement again, we will get, in general, a slightly different value.

We want to make sure that this difference does not affect the results. For this purpose, we want to make sure that:

- if two measurement results are close, i.e., if $x \approx x'$,
- then the corresponding values of the membership function should also be close: $\mu(x) \approx \mu(x')$.

This is exactly what is called robustness.

. . . variational derivative? In variational optimization problems (see), we need to find a function $f(x)$ which is the best – i.e., of which the value of the corresponding criterion $J(f)$ is the smallest (or the largest) possible. Finding a function means finding its value $f(x)$ for all possible inputs x. To find the optimal value $f(x)$, we can differentiate the criterion J with respect to the unknown $f(x)$ – the resulting derivative is what is called variational derivative – and equate this derivative to 0.

. . . variational optimization? In many situations, we want to select a function which is the best – e.g., we want to select a membership function $\mu(x)$ or an "and"-operation $f_{\&}(a, b)$. Such optimization problems, when we select the best function, are known as variational optimization problems.