An Introduction to Fuzzy and Neurofuzzy Systems

M. Brown

December 18, 1996

Contents

1 Introduction

Fuzzy logic and fuzzy systems have recently been receiving a lot of attention, both from the media and scientific community, yet the basic techniques were originally developed in the mid-sixties. In fact, last year marked the 30^{th} anniversary of Professor Zadeh's seminal paper on the subject. Fuzzy logic provides a formalism for implementing expert or heuristic rules on computers, and while this is the main goal in the field of *expert* or *knowledge-based* systems, fuzzy systems have had considerably more success and have been sold in

automobiles, cameras, washing machines, rice cookers, etc. Two years ago, the market for consumer products was estimated to be $2 billion and while the application of neural networks has been very much academically led, research into fuzzy systems only started seriously once they had proved to be a useful engineering tool.

This report will describe the theory behind basic fuzzy logic and investigate how fuzzy systems work. This leads naturally on to *neurofuzzy* systems which attempt to fuse the best points of neural and fuzzy networks into a single system. Throughout this report, the potential limitations of this method will be described as this provides the reader with a greater understanding of how the techniques can be applied.

1.1 History

In addition to Artificial Neural Networks (ANNs), another technology that came into prominence during the mid-sixties and was subject to a large amount of scepticism was *fuzzy logic*. Lofti Zadeh was a professor in the electrical engineering department at UC Berkeley when he first discovered what was to become known as fuzzy logic. He had had a long background in linear systems theory, and had come to the conclusion that engineers and scientists had become overly concerned with the pursuit of precision. In 1973, he formulated this in what became known as the *principle of incompatibility* [38]

> *as the complexity of a system increases, our ability to make precise and yet significant statements about it diminishes until a threshold is reached beyond which precision and significance (or relevance) become almost mutually exclusive characteristics.*

Indeed, he also, at that time, expressed a view that was to hold true when, in the late eighties/early nineties, there was an explosion of fuzzy systems in consumer products:

> *excessive concern with precision has a stultifying influence in control and systems theory, largely because it tends to focus research in this field on those and only those problems which are susceptible to exact solutions.*

Fuzzy logic therefore places the human designer at the centre of the engineering process and in contrast to the *model-based* mathematical techniques that have traditionally been used and the *data-driven* ANNs, fuzzy logic is a *human-centred* design technique.

Zadeh, with a flair for publicity, termed this new technology fuzzy logic, yet the ideas behind graded set membership had a long history, stretching back to the turn of the century when Charles Sanders Peirce (1839-1914), regarded by some as America's most innovative philosopher claimed in 1905 to "have worked out the logic of vagueness with something like completeness". Unfortunately, no technical papers that mark this discovery have ever been found. Vague logic had another prominant proponent in the early twentieth century when Bertrand Russell (1872-1970) [31] claimed that:

> *All language is vague*

and

> *Vagueness, clearly, is a matter of degree*

However, it was Jan Lukasiewicz who proposed the first formal model of vagueness when he introduced three valued logic in 1920 and a host of other famous logicians: Kurt Gödel, John von Neumann, Donald Kleene, Max Black etc., extended this to include multivalued and continuous logics.

Fuzzy, or vague, logic therefore has a history that extends beyond Zadeh's seminal paper in 1965 [37], where he introduced a terminology and expressions which gave the impression that here was a *new* technology. This proved to be almost a killer blow in terms of scientific credibility, as many established academics poured scorn on the ideas postulated by Zadeh, although it did serve to emphasise the fact that a *new* approach to problem solving was being proposed and fuzzy logic was one such algorithm. Incorporating humans directly in the design process means that mechanisms must be found to quantify exactly what is meant by the vague statements made by experts. Whether or not this information source will prove useful is application dependent, but probably the main result from the many applications of fuzzy systems is that human insight and expertise is an important part of *any* design process, and approaches like fuzzy logic which emphasise the role of the expert are crucial in many applications.

In the late sixties, the US funding agencies were actively sponsoring research into expert systems, although these techniques were not accepted until the late seventies when applications such as MYCIN and PROSPECTOR proved that the expert system approach was valid. However, it has been found that a large number of rules are needed for these systems and as such they are difficult to build, validate and maintain, even with the development of advanced expert systems shells. The whole of the Artificial Intelligence (AI) field can be described as a search for a better representation and this is especially true for work into expert systems, where researchers have been searching for better ways of modelling and representing *uncertainty*. Uncertainty can take many forms and in the past, probability and statistical theory has been used to model different types of uncertainty. However, fuzzy logic has been widely applied to modelling *linguistic uncertainty*, and it is only recently that probability theorists have tried to apply their methods in this domain. Fuzzy systems have proved to be simple to develop and the first application of this technology in control by Professor Abe Mamdani and his PhD student Sedrak Assilian in 1974 took just over a weekend to build.

1.2 Uncertainty and Natural Language Vagueness

Fuzzy logic allows an element to be a *partial member* of a set, so its membership value can lie between 0 and 1 and can be interpreted as:

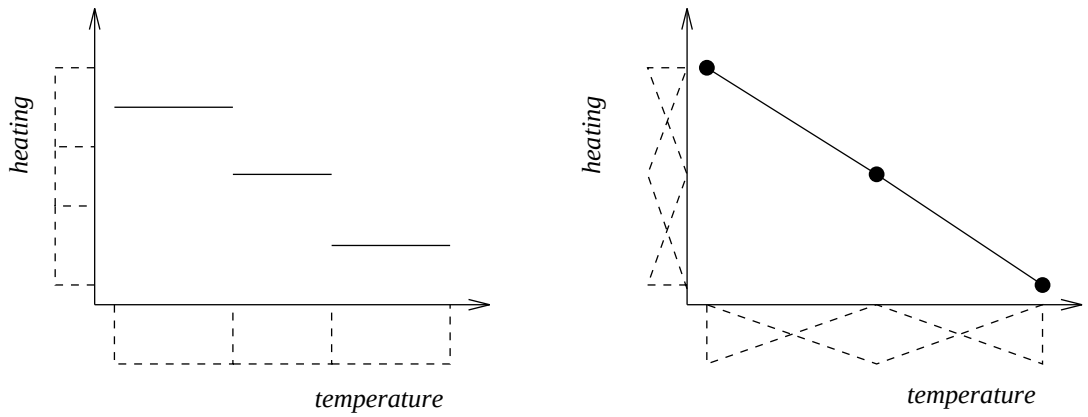
> *the degree to which an event may be classified as something.*

It allows elements to be members of different sets with varying degrees at the same time, and also allows ordering information to be retained in the class membership values. Yet it is important to realise that the uncertainty is not an inherent property of the event; rather it comes about because of the classification system. This is very important in the area of expert systems, as natural language terms such as *small* and *hot* are imprecise or vague yet humans reason and convey useful information using such terms, expecting a system to be able to *generalise* between neighbouring concepts. Rules such as:

IF (*the room is hot*) THEN (*turn down the heating*)
OR IF (*the room is warm*) THEN (*keep the heating constant*)
OR IF (*the room is cold*) THEN (*turn up the heating*)

involve natural language statements which should not be modelled using conventional sets as this produces the result illustrated in figure 1a. Fuzzy sets are *graded*, which allows the system to generalise or interpolate between rules as shown in figure 1b.

Generalisation or interpolation between concepts/rules is taken for granted in much of natural language. However this is *not* generally modelled in conventional expert systems,



(a) (b)

Figure 1: The output of a binary (a) and a fuzzy (b) expert system.

and fuzzy logic with its graded or multivalued set membership, is a better representation for this type of vagueness or uncertainty. It is important to realise that simply adopting a fuzzy or multivalued set membership scheme is *not* sufficient to completely represent *all* natural language statements. as using fuzzy, or multivalued, logic in order to represent expert knowledge is necessary but not in general sufficient for this problem.

1.3 Information Representation

When data are presented to a network (neural, fuzzy, etc.) they can be classified according to the type of information they contain.

Nominal variables refer to quantities for which no ordering relationships exist between the elements and the only tests which can be performed are equality and inequality. An example of this is the set:

$$fruit = \{apple,\ pear,\ banana,\ carrot\}\ .$$

This is how elements are represented in a conventional set where they are either a member or not a member, and no other information is contained in the set membership value.

Ordinal variables refer to quantities where an ordering relationship also exists between the individual elements. An example of this is the set intelligent:

$$intelligent = \{Mick,\ Sue,\ Rita,\ Bob\}$$

where

$$intelligence(Bob) > intelligence(Sue) > intelligence(Rita) > intelligence(Mick).$$

The fact that Bob is more intelligent than $Mick$ must be reflected in the value of the set memberships. Bob is the more intelligent, hence his qualities must be closer to the ideal definition of "intelligence", and this implies that representing vagueness is strongly related to the ideas of expressing abstract distances from a set's ideal. Fuzzy sets employ graded set membership and as such its output is an ordinal variable.

Interval variables provide a richer description than ordinal variables, as the difference between variables can be interpreted and ranked. Therefore, the differences in intelligence for the members of the previous (fuzzy) set could be expressed as:

$$intelligence(Bob) - intelligence(Sue) > intelligence(Rita) - intelligence(Mick)$$

Fuzzy sets can be designed to incorporate this interval-type knowledge, and it is up to the designer and user to see that it is incorporated correctly.

Ratio variables are like interval variables except that the elements are measured with respect to an absolute scale. This means that it is correct to say that x is twice as hot as y (measured with respect to Kelvin) because $x/y = 2$, or Bob is twice as intelligent as $Mick$ because $intelligence(Bob)/intelligence(Mick) = 2$. Again, fuzzy sets can be designed to incorporate this ratio-type information, but they must be carefully designed and used.

1.4 Fuzzy Representation

Fuzzy sets can therefore represent two types of information which a conventional set is unable to. This means that they provide a richer representation which is potentially closer to the way that humans use vague, natural language knowledge.

Graded set membership allows the value of the set membership to be interpreted as meaning that some elements are more representative of the concept than other elements. The relationship $intelligence(Bob) > intelligence(Mick)$ implies that Bob is more representative of the concept of intelligence than $Mick$ and the relative values can be used to infer information about the relative degree of intelligence.

Multi-set membership allows an element to be a partial member of two or more classes, and also the value of the membership can be used to infer which class is most representative of that element, if necessary. For instance, in the previous definition of the class fruit, a carrot was included as a member! This is a real-world example as they manufacture carrot jam in Portugal and current EU regulations specify that you can have only fruit jam, implying that a carrot is a fruit. However, it is common-sense that a carrot is a vegetable, and so a more sensible, richer classification scheme would be:

$$fruit(carrot) = 0.1$$
$$vegetable(carrot) = 0.9$$

which reflects the fact that a carrot can be classified as either *fruit* or a *vegetable*, but it is more representative of the latter concept. In the real-world most sets are non-mutually exclusive, and an appropriate set theory should be able to model this appropriately.

1.5 Current Information

Research into fuzzy logic and neurofuzzy systems is currently undergoing something of a renaissance, as people are integrating the learning abilities of neural networks with the rule-based representation of fuzzy systems, and the interested reader may find extra information about this topical subject in the following places.

Currently, there are many books being published in the area of fuzzy logic and fuzzy control, the simplest being a non-technical book [25] which describes the development of fuzzy logic research and applications. Introductory technical books include [7, 9, 12, 39], and there are several research books out that describe the current state of the art in fuzzy and neurofuzzy systems theory [1, 5, 21, 28, 34]. There are also two main journals in this area:

where the former is published by North Holland (Amsterdam). Many other AI and engineering journals frequently publish papers about fuzzy logic.

A subscription to the largest fuzzy mailing list can be obtained by sending a message to:

```
listproc@vexpert.dbai.tuwien.ac.at
```

with the following lines of text:

```
help
subscribe fuzzy-mail YourName
```

This is a very active mailing list that contains information about contacts in the field and sometimes has extremely useful discussions about the exact nature of fuzzy logic, fuzzy control etc.

A useful source of current information about share/free and commercial software, hardware, mailing lists, homepages, conferences etc., is maintained in the Image, Speech and Intelligent Systems (ISIS) research group web entry at:

```
http://www-isis.ecs.soton.ac.uk/research/nfinfo/fuzzy.html
```

and a European network for uncertainty modelling and fuzzy technology has a homepage at:

```
http://www.mitgmbh.de/erudit/
```

This is being led by Prof Zimmermann's group at Aachen:

```
http://www.mitgmbh.de/elite/elite.html
```

who maintain a commercial database which contains details about neural and fuzzy research. The Berkeley Initiative into Soft Computing (BISC), of which Lofti Zadeh is the director, has a web homepage at:

```
http://http.cs.berkeley.edu/projects/Bisc/bisc.welcome.html
```

which includes information about the centre itself, links with other homepages and contact information with many leaders in the field of soft computing as well as employment openings. Also, the North American Fuzzy Information Processing Society (NAFIPS) web homepage is located at:

```
http://serphim.csee.usf.edu/nafips.html
```

Finally, the address of the fuzzy newsgroup is:

```
comp.ai.fuzzy
```

2 Fuzzy Set Theory

One of the main reasons for using static fuzzy logic[1] is to exploit any available vague expert knowledge in a computer programme. An expert's knowledge is frequently expressed in terms of a *fuzzy algorithm* which is composed of IF-THEN production rules, that relate vague input statements to vague output actions or states. For instance, the following two rules might form part of a larger rule base in a fuzzy expert system that tries to model human behaviour:

IF (*Martin is hungry*) THEN (*eat a snack*)
OR IF (*Martin is starving*) THEN (*eat a large dinner*)

The terms in the rules' antecedents (the input part of the rule) specify imprecisely the state of well-being of a fictional character called Martin and the consequents represent courses of action which could be taken to rectify these undesirable events.

Fuzzy sets give the designer a method for providing a *precise* representation of vague, natural language terms such as *hungry, starving, large etc.* Fuzzy logic provides the necessary mechanisms for manipulating and inferring conclusions based on this vague information. In this report, we will look at how this is performed in greater detail, from the choice of the fuzzy membership functions to the operators that are used to implement intersection, union etc., and try to analyse what is happening inside the fuzzy system. It is perhaps one of the great fallacies that all fuzzy systems are transparent to the designer[2], as the type of decision surfaces formed are generally poorly understood, and the fuzzy rules give the designer only a partial insight into its internal structure.

2.1 Classical Sets

Fuzzy sets were introduced to overcome some of the limitations of classical (boolean or binary) sets. Classical set theory has been extremely useful, and it now underpins much of the theory of mathematics, although when practitioners tried to apply it to real-world objects and events, they kept coming across a phenomenon that has been termed the *Sorites paradox.* This can be explained as:

> *When does a heap of sand stop being a heap of sand if you keep taking one grain away?*

Classical sets introduce a threshold value which specifies exactly how many grains of sand constitute a heap, as shown in figure 2a.

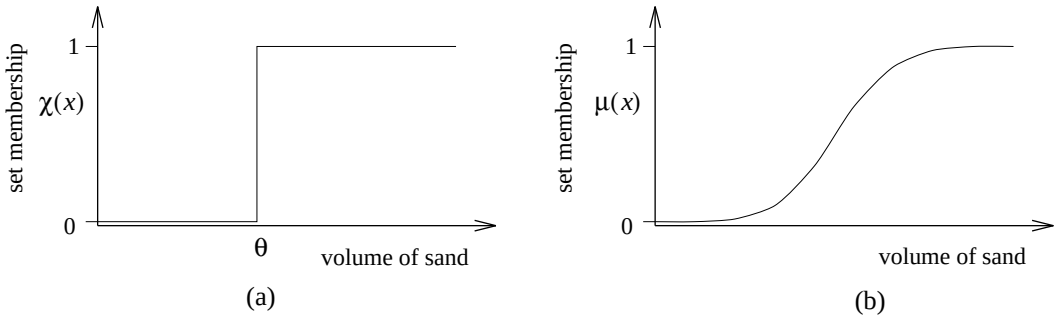


Figure 2: A classical set (a) representation of *sand heap* as well as a fuzzy one (b).

[1] Adaptive fuzzy systems form their own rule base.
[2] A system is transparent if its internal workings can be easily understood.

A classical set is generally represented by listing its set of members, or else by giving a concise functional relationship which its members must satisfy. For example, the two sets *greater than 3*, and *cars* would be represented as:

$$\begin{aligned} \textit{greater than 3} &= \{x \in \Re : x > 3\} \\ \textit{cars} &= \{\text{Mini, Rover, Ferrari, } \ldots\} \end{aligned}$$

Note that for the first example, the set of possible inputs is infinite and so it is represented as a mathematical expression whereas for the second, a finite list is used.

It is also possible to define a set in terms of whether or not an element is a member of that set, and this is determined by the *characteristic function*. Suppose that the space on which the elements of a set are defined is called the *universe of discourse* (this could be the real line, or the set of all objects in the world, or a particular input measurement), then the characteristic function is defined by:

Definition 2.1 (Characteristic Function) *The classical set A, which is defined on its universe of discourse X, is defined by its* characteristic function $\chi_A() : X \to \{0, 1\}$ *which maps each element of X to either 1 or 0 depending on whether or not that element is a member of the set. This is represented as:*

$$\chi_A(x) = \begin{cases} 1 & \textit{if } x \in A \\ 0 & \textit{otherwise} \end{cases} \tag{1}$$

For the sand heap example, the characteristic function would have the form:

$$\chi_{heap}(x) = \begin{cases} 1 & \text{if } x > \theta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where x is the volume of sand in the pile and θ is the value of the threshold. A subtle but often overlooked point is that characterising a pile of sand as a heap would also depend on its shape, as well as other factors. However, humans have a remarkable tendency to round-off details that may be important in certain situations. Similarly, the value of a threshold would not be constant for different situations, rather it would depend on the context in which it is used. Classical set theory is extremely well understood, but a major problem occurs with its interface to the real-world.

This binary representation of the concept of a heap of sand is unable to represent the transition process from *heap* to *not heap*, as illustrated by the Sorites paradox. This does not mean that boolean sets are an inappropriate representation for every real-world set, rather the introduction of a threshold value means that a lot of *ordering* information (ordinal variables) is lost.

The fuzzy set characterisation of the concept of a heap of sand is shown in figure 2b, and it can be clearly seen that the idea of partial membership of a set allows the designer to represent a gradual ordering process as the degree of membership can now take any value in the unit interval. It may be that in the decision making process, a threshold value will have to be applied, but this process can be delayed as no information has been lost in representing the input variable as a (fuzzy) set membership value, as in this case the fuzzy membership function is invertible.

2.2 Fuzzy Membership Functions

Just as a classical set is defined by its characteristic function, a fuzzy set is represented by its *membership function* $\mu(.) \in [0, 1]$. This means that an element can be a partial member of a particular set, and as mentioned in the introduction, this can be viewed as representing:

The degree to which an event may be classified as something.

It does *not* measure the frequency of occurrence of an event (that would be relative probability) or necessarily an individual's uncertainty about an event (subjective probability), rather it allows real-world events to be classified using a finite number of linguistic classes.

Imagine that a Remote Sensing system was being developed where the maximum sensor resolution was a kilometer square. Within that land area, the land-usage may be partially urban, partially green space and partially agriculture. The measurement is exact, the classes {*urban, green space, agriculture*} are well-specified, but just about every measurement should produce a classification which is composed of a combination of land-usage. Developing a statistical (maximum-likelihood) classifier would produce an answer which represented the class with the highest land usage, whereas the true picture is that each event (pixel-value) corresponds to a (non-zero) membership in several classes.

Generally, fuzzy membership functions are defined on just one variable or measurement (univariate), but this need not be the case as we shall see in the following sections where logical connectives are used to generate fuzzy membership functions defined on several variables or measurements (multivariate) from the standard univariate ones.

Definition 2.2 (Fuzzy Membership Function) *The fuzzy* membership function *of a set A is defined on its universe of discourse X and is characterised by the function $\mu_A(.) : X \to [0, 1]$ which maps each element of X to a real number that lies in the unit interval $[0, 1]$. For a particular input, the value of the fuzzy membership function represents the degree of membership of that set.*

The membership functions provide an *interface* between the real-valued feature (input) space and an expert's vague, linguistic sets. They form *precise* representations of vague concepts, but this precision is useful because it contains an expert's domain specific knowledge about a particular situation. If you need to model the imprecision associated with a particular vague set, probability or rough set theory may be used to model the variance in the set's shape, but in most successful fuzzy systems the standard membership function has proved to be sufficiently useful.

This idea of graded set membership can be used to represent the concept of a *heap*, yet still retain the important ordering information. When $\mu_{heap}(x)$ is slightly greater than $\mu_{heap}(y)$, this would imply that the volume of sand x is only a little more than y, and so this important quantitative ordering information is retained in the fuzzy set representation.

A single fuzzy set by itself can model ordering information, but the main reason for employing fuzzy logic and developing fuzzy systems is to infer decisions and information from several pieces of expert knowledge. Humans learn to build up a finite number of appropriate categories for describing a potentially infinite number of real world events and because we communicate using *similar* vocabularies, vague but important information can be communicated. For the vast majority of applications, the power of a fuzzy system comes about because of its ability to generalise and as such the definition of one fuzzy set is related the definition of its neighbouring ones. The definition of a fuzzy set is *relative* to how its neighbours are defined.

2.2.1 Terminology

Definition 2.3 (Crisp) *The fuzzy membership function $\mu_A(.)$ is said to be* crisp *if:*

$$\mu_A(x) \in \{0, 1\} \qquad \forall\, x \in X \tag{3}$$

and this is illustrated in figure 3a. A crisp fuzzy set is therefore just a conventional set and under this restriction, a fuzzy reasoning system often employs standard boolean logic. This terminology may initially seem unnecessary but some real-world concepts are crisp, and if fuzzy logic is to be seen as a complete system, it should be able to model this appropriately.

Definition 2.4 (Singleton) *The fuzzy membership function* $\mu_{\tilde{x}}(.)$ *is referred to as a* singleton *when:*

$$\mu_{\tilde{x}}(x) = \begin{cases} 1 & \text{if } x = \tilde{x} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

It follows therefore that a singleton fuzzy set is a type of crisp set for which it is non-zero only at a single input value, as shown in figure 3b.

Definition 2.5 (Unimodal) *The fuzzy membership function* $\mu_A(.)$ *is said to be* unimodal, *if:*

$$\forall x, y \in X,\ \forall \lambda \in [0,1] :\ \mu_A(\lambda x + (1-\lambda)y) \geq \min\{\mu_A(x), \mu_A(y)\} \tag{5}$$

A unimodal fuzzy membership function is often known as *convex*, but the author prefers the former terminology as it is more descriptive of membership function's shape (see figure 3c). The vast majority of fuzzy membership functions are unimodal, and while all of the fuzzy theory described in the following sections is true for any shaped set, we shall be only concerned with unimodal ones. A unimodal membership function implies that the linguistic term only has a *local* (or one-sided) influence on the overall rule base. It is difficult for humans to understand how large numbers of rules interact, so this is an important property for a fuzzy system if it is required to be transparent to the designer.

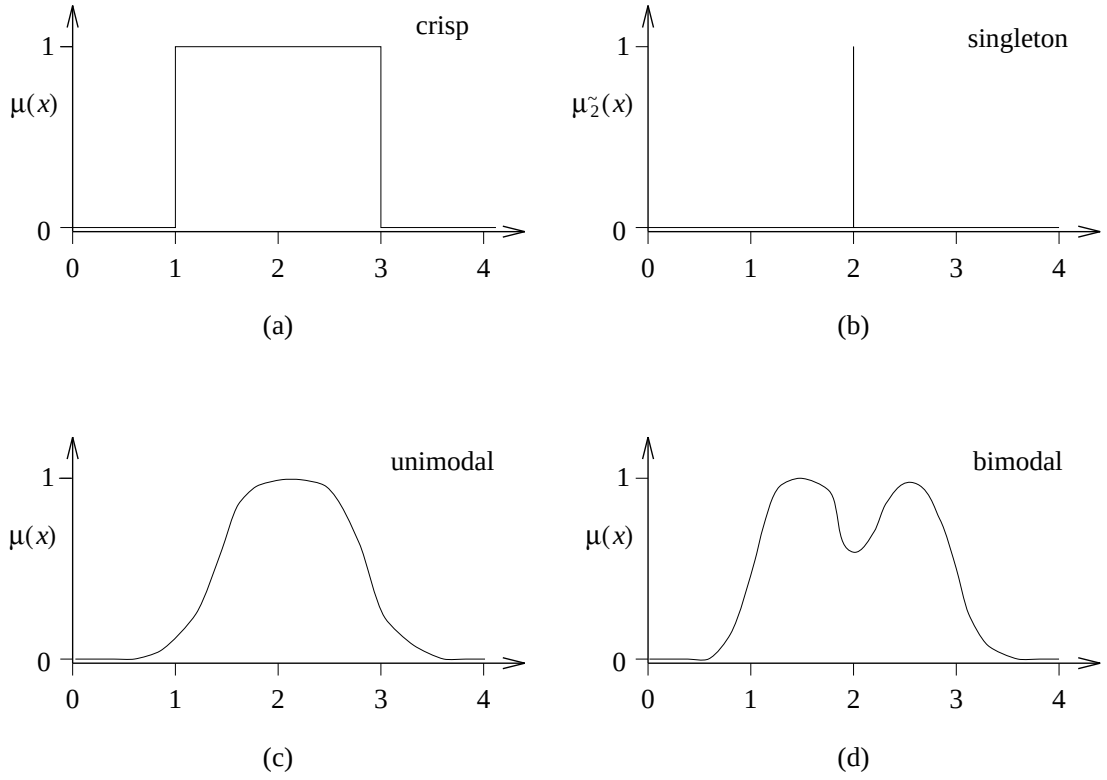


Figure 3: An illustration of a crisp fuzzy set (a), a singleton fuzzy set centred on the input 2 (b), a unimodal (c) and a bimodal (d) fuzzy membership function.

Definition 2.6 (Support) *The* support *of a fuzzy set S_A is given by the following set:*

$$S_A = \{x \in X : \mu_A(x) > 0\} \tag{6}$$

The support of a fuzzy set A is therefore the part of the input space for which its membership function is activated to a degree greater than zero. An important, related concept is when a membership function has a *compact* support. Compactness refers to the fact that the size of its support is strictly less than the size of the original universe of discourse, and this is illustrated in figure 4. The support of a fuzzy membership function therefore determines

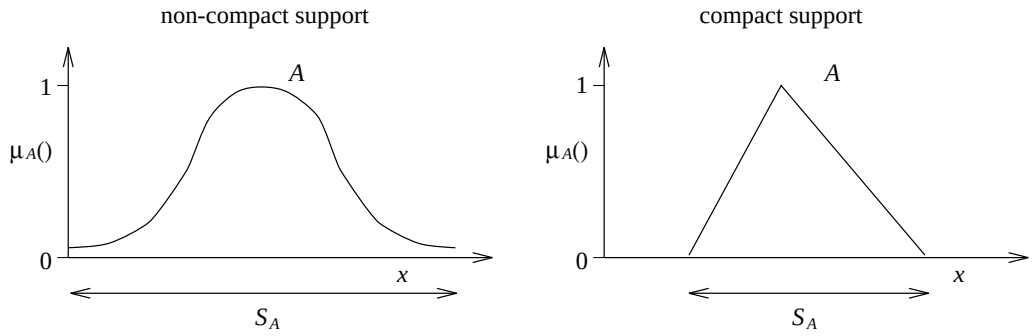


Figure 4: A non-compact support and a compact support fuzzy set.

which inputs will activate fuzzy rules that have the corresponding linguistic term as part of their antecedent. If the fuzzy membership functions have a non-compact support then every rule will be activated by each input, and the important concept of local knowledge storage and retrieval may be lost.

Definition 2.7 *The* α-cut *of a fuzzy set A is given by the membership function:*

$$\mu_{A^\alpha}(x) = \begin{cases} \mu_A(x) & \text{when } \mu_A(x) \geq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\alpha \in [0, 1]$.

Hence any unimodal membership function with a non-compact support can be transformed into a (discontinuous) membership function by taking the appropriate α-cut, as illustrated in figure 5.

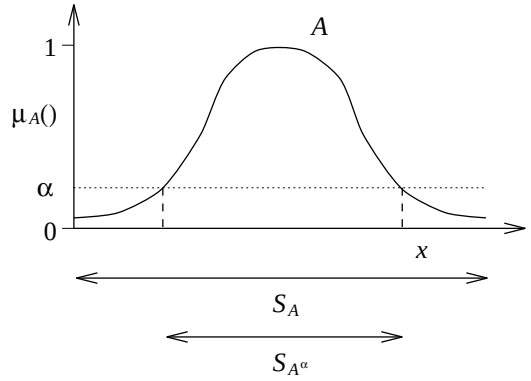


Figure 5: Taking the α-cut of a set with a non-compact support.

Definition 2.8 *The* height *of a fuzzy set A is defined as:*

$$H_A = \max_{x \in X} \{\mu_A(x)\} \tag{8}$$

and a set is known as *normal* if $H_A = 1$, and *sub-normal* otherwise.

Definition 2.9 *The fuzzy set A is known as a* fuzzy number *when it is normal and defined on the real-line.*

It is important to notice that *all* of these concepts are local to one particular fuzzy set, whereas the power of a fuzzy system comes through its ability to generalise locally between neighbouring sets and rules. Probably the main use of fuzzy sets is to locally interpolate or extrapolate between several rules in a fuzzy system, hence, it is difficult to say whether a particular membership function is well-designed *without* reference to the remaining ones in the fuzzy system. This point will be re-emphasised in sections 2.2.3 and 3.2.

2.2.2 Distance Measures and Fuzzy Membership Functions

A unimodal fuzzy membership function contains ordering information such that when:

$$\mu_A(x) > \mu_A(y) \tag{9}$$

for a particular fuzzy set A, we can interpret this as meaning that x is "closer" to the ideal definition of A than y. Thus fuzzy sets and distance measures are synonymous as the membership function $\mu_A(x)$ and a distance measure $d(A, x)$ are related by:

$$d(A, x) = \begin{cases} \infty & \text{if } \mu_A(x) = 0 \\ \frac{1}{\mu_A(x)} - 1 & \text{otherwise} \end{cases} \tag{10}$$

as illustrated in figure 6. When a fuzzy set is designed, its *centre*[3] represents a *template* of

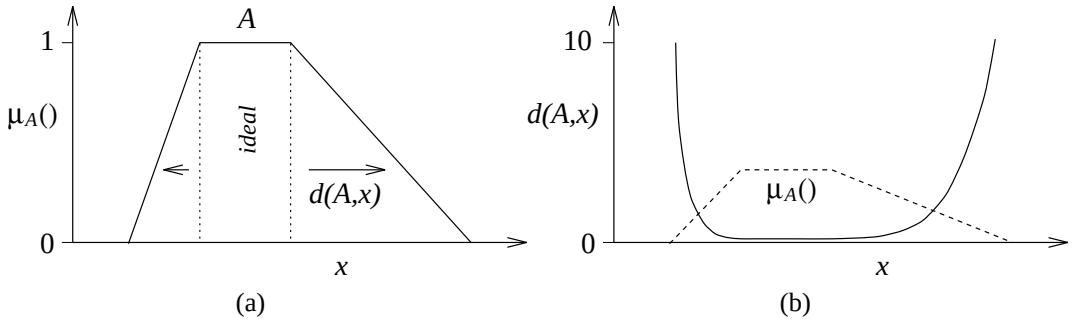


Figure 6: The fuzzy set A and its associated distance measure $d(A, x)$ and vice versa.

the definition or the set of ideal members and the tail off determines how this set interacts with others in a complete fuzzy rule base.

The inverse relationship can also (obviously) be used as a formal definition of a fuzzy membership function if the set's sensitivity is defined in terms of a distance measure. This produces:

$$\mu_A(x) = \begin{cases} 1 & \text{if } d(A, x) = 0 \\ \frac{1}{d(A,x)} + 1 & \text{otherwise} \end{cases} \tag{11}$$

[3]The centre of a fuzzy set is defined as being those elements which activate it to membership 1.

2.2.3 Fuzzy Linguistic Variables

In order to describe or represent a real-world measurement as a symbolic or fuzzy label, it is necessary to define two or more fuzzy sets on that particular variable. For instance, the temperature of the water may be described as *cold*, *warm* or *hot*, and the definition of each term is *relative* to the neighbouring sets. Once the number of linguistic terms that are used to model a variable have been decided, and their form has been specified, the complete set of fuzzy membership functions is known as a *fuzzy variable*. Describing a variable using a single linguistic symbol does not provide any more information than simply ignoring that variable from the inferencing calculations. If a variable is always classified as being *hot*, we don't need to measure the temperature as this is known *a priori* and can be implicitly incorporated into the rule base. Therefore, the power of fuzzy sets and fuzzy system is due to the *relative* definition of each of the linguistic terms.

Loosely speaking, a fuzzy variable V_X is formed when a group of fuzzy membership functions and their corresponding linguistic terms are associated with a particular variable. A fuzzy set A, is uniquely associated with its universe of discourse, so a fuzzy variable can be imagined as being the group of all the fuzzy sets associated with this particular variable. More precisely:

Definition 2.10 (Fuzzy Variable) *A fuzzy variable V_X is defined as the 4-tuple: $\{X, L, \chi, M_X\}$, where:*

- *X is the symbolic name of a linguistic variable, such as* age.
- *L is the set of linguistic labels associated with X such as* {young, middle aged, old}.
- *χ is the domain over which L is defined on the universe of discourse of X. For a real-valued variable, this could be a continuous interval such as $[0, 120]$ (years) or a discrete, sampled set such as $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$.*
- *M_X is a semantic function that returns the meaning of a given linguistic label in terms of the elements of X and the corresponding values of the fuzzy membership function.*

A fuzzy variable therefore is a collection of all the information used to represent a particular measurement as a fuzzy linguistic set.

Example 2.1 (Fuzzy Variable) *Consider designing a one-touch grill for barbecueing sausages where the symbolic name of the linguistic variable is $X =$* cooking time *(in minutes) and the linguistic term set defined on this variable is $L =$* {rare, medium, well-done, charcoaled}. *Then the real domain over which X is defined could be from 2 to 30 minutes and the semantic functions which return the membership functions are illustrated in figure 7.*

Fuzzy variables are very important in the overall system, as a single fuzzy membership function and rule by itself is often *no* richer than a single crisp rule. This may initially seem counterintuitive, but consider the following (fuzzy) rule:

> IF (*medium pressure is applied to the throttle*)
> THEN (*the car goes fast*)

and the result of being presented with the following pieces of evidence:

> *a small amount of pressure is applied*
> *a large amount of pressure is applied*

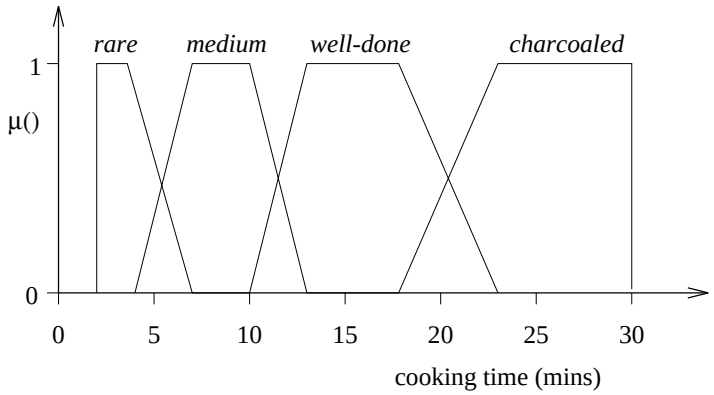


Figure 7: A fuzzy variable corresponding to the length of cooking time for sausages on a barbecue.

Intuitively, we know that the former would need an output less than that contained in the original rule and the latter should produce an output which is greater. However, this information cannot be contained in a single fuzzy set, and it is its neighbouring sets (and rules) which describe how a throttle is pressed which determine how this set affects the overall system. Indeed, using simple fuzzy inferencing operations would cause the system to produce an output that was the same as the original rule. Hence, it is impossible to consider how a single set influences a system's performance without looking at the rules associated with the complete fuzzy variable.

Just as a single fuzzy membership function has certain properties (unimodal, compact support etc.), there are a number of important properties that need to be assessed about each fuzzy variable that is defined.

Definition 2.11 (Completeness) *The fuzzy variable V_X is said to be* complete *if for each $x \in X$ there exists a fuzzy set A such that:*

$$\mu_A(x) > 0 \tag{12}$$

Obviously, when a fuzzy variable is not complete, there exist inputs which have no linguistic interpretation in terms of the current term set and hence the output of *any* system that is based on these linguistic sets will be zero (undefined). A useful, related concept is that of *α-completeness* where the α-cut membership functions are tested for completeness. This will provide the designer with some measures about how well the fuzzy membership functions cover the universe of discourse.

Definition 2.12 (Partition of Unity) *The fuzzy variable forms a* partition of unity *(sometimes known as a fuzzy partition) if for each input x:*

$$\sum_{i=1}^{p} \mu_{A^i}(x) \equiv 1 \tag{13}$$

Obviously, this is a stronger condition than completeness, and a sufficient condition for completeness is that the fuzzy variable forms a partition of unity. Requiring a fuzzy variable to form a partition of unity is a restrictive condition and is similar to one of the fundamental axioms of probability theory where it is required that the sum over the probabilities always equals one. However, for many engineering applications, it can be shown that when the designer incorporates this property into the overall fuzzy system, the system is more transparent

and its performance is improved. Indeed, the final operation in many fuzzy systems is a type of normalisation calculation and any fuzzy variables that do not form a partition of unity have this property implicitly imposed on them. Therefore, it gives the system designer much greater control to impose this condition explicitly on the fuzzy membership functions, prior to the reasoning and inferencing calculations, and this is discussed further in section 2.3.3.

2.3 Types of Fuzzy Membership Functions

The actual shape of the fuzzy membership functions that are used to represent the linguistic terms are relative, subjective and context dependent and as such are ill-defined. However, the basic form should satisfy the following two points.

- It must broadly possess the properties that are representative of the fuzzy linguistic terms. For instance, it may be required that the membership functions are unimodal, have a compact support and form a partition of unity.
- The membership functions must have a simple representation so that their form can easily be stored in a computer's memory and to ensure that the membership of a particular input can be quickly and accurately evaluated.

Unlike probability theory, where a unique probability density function is determined by the statistics of the signal, a fuzzy membership function by its very nature is extremely difficult to determine precisely. Therefore, simple shapes such as the Gaussian bell curve or the piecewise polynomial B-splines are often used to represent the fuzzy membership functions or else they are learnt directly from some training data.

2.3.1 B-spline Membership Functions

B-spline basis functions are piecewise polynomials of order k, which have been widely used in surface fitting applications, but they also can be used as a technique for designing fuzzy variables. B-splines basis functions are defined on a (univariate) real-valued measurement and are parameterised by the *order* of the piecewise polynomial k and also by the *knot vector* which is simply a set of values defined on the real-line that break it up into a number of intervals. This information is sufficient to specify a set of basis (membership) functions defined on the real-line whose shape is determined by the order k and where each membership function has a compact support k units wide. In addition, the set of membership functions form a partition of unity (see definition 2.12). The different shapes of membership functions, for different values of k, are shown in figure 8, and it can be seen that they can be used to implement binary, *crisp* fuzzy sets ($k = 1$) or the standard triangular fuzzy membership functions ($k = 2$) as well as smoother representations [5, 6, 30]. The B-spline basis function therefore provide the designer with a flexible set of fuzzy set shapes, all of which can be evaluated efficiently.

As well as choosing the shape (or order) of the membership functions, the designer must also supply a knot vector which determines how the membership functions are defined on its universe of discourse. Suppose there exist p linguistic terms (and hence fuzzy membership functions) in the fuzzy variable, then the membership functions are defined on an interior space $p - k + 1$ intervals wide and the designer must specify $p + k$ knot values, λ_i, which satisfy the following relationship:

$$x_{\min} < \lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_{p-k} < x_{\max}. \tag{14}$$

These knots roughly correspond to the centres of the individual basis functions, and as such can be used to distribute them such that there is a fine resolution (large number of

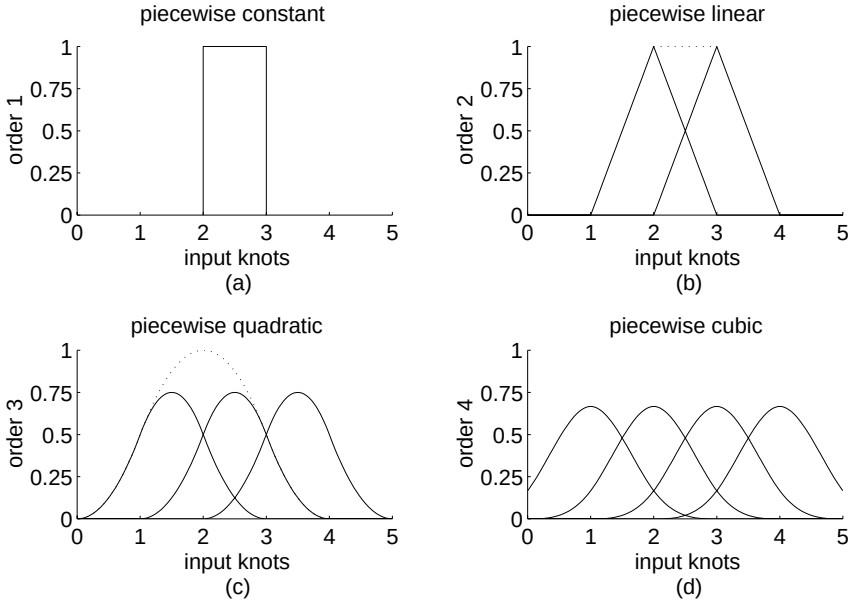


Figure 8: B-spline fuzzy membership functions of orders $1-4$. The dotted lines show how trapezoidal and Π fuzzy membership functions can be formed from an *additive* combination of piecewise linear and quadratic basis functions, respectively.

basis functions) in areas of interest and a course resolution (small number of basis functions otherwise. A set of the extrema knot values which define the basis functions at each end must also be specified and these should satisfy:

$$\lambda_{-k+1} \leq \cdots \leq \lambda_0 = x_{\min} \tag{15}$$

$$x_{\max} = \lambda_{p-k+1} \leq \cdots \leq \lambda_p \tag{16}$$

and this is illustrated in figure 9 for order 2 (triangular) fuzzy membership functions.

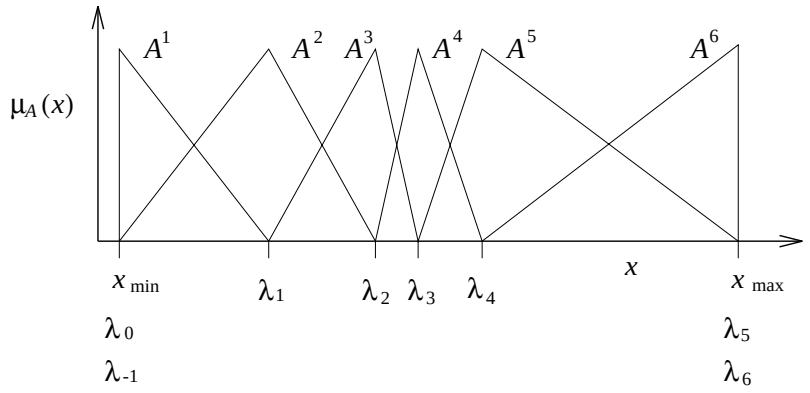


Figure 9: Six B-spline fuzzy membership functions of order $k = 2$ where a non-uniform knot placement strategy is used.

The output of B-spline membership functions can also be calculated using the following simple and stable recurrence relationship:

$$\mu_{A_j^k}(x) = \left(\frac{x - \lambda_{j-k}}{\lambda_{j-1} - \lambda_{j-k}}\right) \mu_{A_{j-1}^{k-1}}(x) + \left(\frac{\lambda_j - x}{\lambda_j - \lambda_{j-k+1}}\right) \mu_{A_j^{k-1}}(x)$$

$$\mu_{A_j^1}(x) = \begin{cases} 1 & \text{if } x \in [\lambda_{j-1}, \lambda_j) \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

where $\mu_{A_j^k}(x)$ is the j^{th} membership function of order k. Therefore using B-spline basis functions as a framework for fuzzy membership functions has several important properties:

- A simple and stable recurrence relationship can be used to evaluate the degree of membership.
- The basis functions have a compact support which means that knowledge is stored locally across only a small number of basis functions.
- The basis functions form a *partition of unity* which also implies that the corresponding fuzzy variable is *complete*.

Many of the piecewise polynomial fuzzy membership functions that have been used in the literature are simply particular types of standard, additive or dilated B-splines [16, 23].

2.3.2 Gaussian Membership Functions

Another fuzzy membership function that is often used to represent vague, linguistic terms is the *Gaussian* which is given by:

$$\mu_{A^i}(x) = \exp\left(-\frac{(c_i - x)^2}{2\sigma_i^2}\right) \tag{18}$$

where c_i and σ_i are the centre and width of the i^{th} fuzzy set A^i, respectively. This is illustrated in figure 10. Gaussian fuzzy sets have some very desirable properties in that both their spatial and frequency content (a Fourier transform of an exponential is another exponential) is local, although not strictly compact, and the output is very smooth in that it can be differentiated as many times as you like. However, it is important to note that although these functions can also represent (scaled) probability density functions, their meaning in this context is generally different. Probability density functions can be used to calculate the probability (or relative frequency) and a measurement will lie in an interval, whereas fuzzy set theory provides a measurement of the degree of membership that an exact measurement satisfies a vague concept. Gaussian functions can be used to model both situations but the underlying meaning is very different.

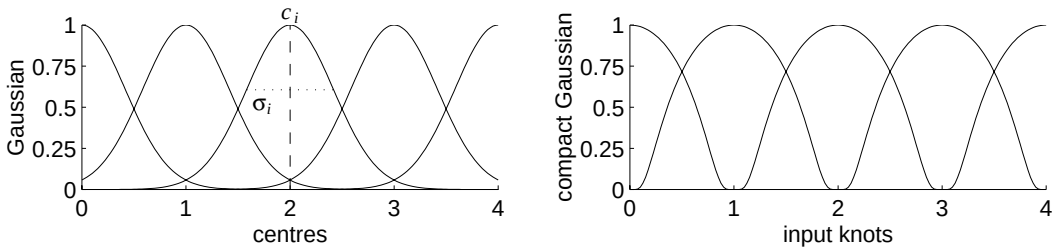


Figure 10: Gaussian fuzzy sets.

Multivariate Gaussian functions are formed from the *product* of the univariate sets, and this is one of the reasons why Gaussian fuzzy sets are popular; the multivariate radial basis functions can be expressed as a product of univariate ones. This emphasises the relationship between fuzzy sets and distance measures. An interesting *compact support* Gaussian-type

function has also been proposed [35], and this membership function has a strictly compact support property and is also infinitely differentiable:

$$\mu_{A^i}(x) = \begin{cases} \exp^{-1}(-1) * \exp\left(-\frac{(\lambda_{i,2}-\lambda_{i,1})^2/4}{(\lambda_{i,2}-x)(x-\lambda_{i,1})}\right) & \text{if } x \in (\lambda_{i,1}, \lambda_{i,2}) \\ 0 & \text{otherwise} \end{cases}$$

and this is also shown in figure 10.

Gaussian fuzzy membership functions are quite popular in the fuzzy logic literature, as they are the basis for the link between fuzzy systems and Radial Basis Function (RBF) neural networks.

2.3.3 Partitions of Unity and Fuzzy Variables

It is worthwhile emphasising that *any* complete fuzzy variable can be transformed into a fuzzy variable that forms a partition of unity by normalising the membership functions according to:

$$\mu_{\hat{A}^i}(x) = \frac{\mu_{A^i}(x)}{\sum_{j=1}^{p} \mu_{A^j}(x)} \qquad \text{for each } i = 1, \ldots, p. \tag{19}$$

as by definition, the modified fuzzy membership functions form a partition of unity:

$$\mu_{\hat{A}^i}(x) \equiv 1 \qquad \forall\, x$$

This is illustrated in figure 11 for Gaussian functions where the widths have been incorrectly set. In general, the normalised version will produce a much smoother output surface, as it does *not* depend on the variation of the activation strength (defined as the sum over all the fuzzy membership functions for a particular input value). It also makes the form of the output surface partially invariant to changes in the location and scaling of the fuzzy membership functions. This can easily be seen as the term $\sum_{i=1}^{p} \mu_{A^i}(x)$ is plotted in this figure as a dashed line which represents the strength with which various rules will fire. In general, it is undesirable for this term to differ significantly from unity, although in practice this is difficult to achieve unless either the fuzzy variable forms a partition of unity implicitly or it is explicitly normalised [5, 27, 35].

2.4 Linguistic Vagueness and Fuzzy Precision

Fuzzy membership functions provide a *precise* representation of linguistic vagueness. This may initially seem at odds with the original reason for adopting this technique, because, as the name suggests, fuzzy logic should utilise an imprecise problem solving approach. However, to implement an algorithm on a computer requires that all imprecision and uncertainty is resolved and modelled precisely (consider modelling the noise processes in a Kalman filter or representing a probability density function as a normal or Poisson distribution). Often it is the *ordering* information ($\mu_A(x) > \mu_A(z)$) that is important rather than the exact value of the fuzzy membership functions. However, the precise membership functions can be useful for extracting an individual's subjective view and context dependent information about the problem. Letting several experts design their own membership functions and rule bases may provide the system designer with important information about different strategies for implementing the system. Similarly, a single expert's knowledge is useful precisely because it is subjective, so clarifying any vagueness using precise membership functions is the first part of any knowledge elicitation strategy.

Natural language is notoriously *rich*, in the sense that a single phrase can have many meanings and there are many ways for expressing the same idea. A well-defined fuzzy system

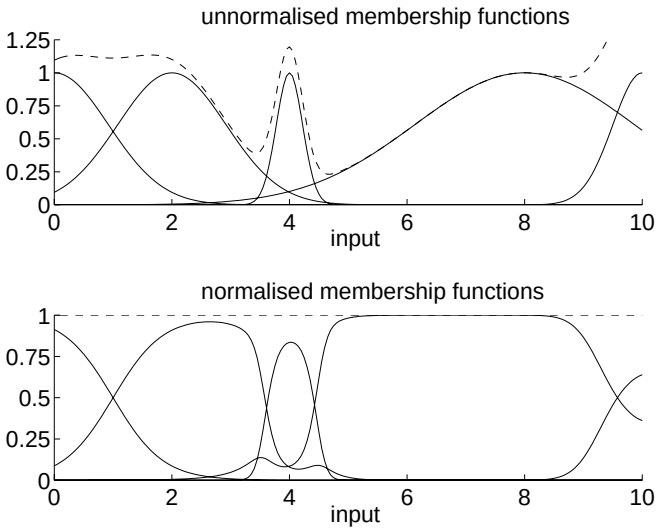


Figure 11: Gaussian membership functions (top) and the equivalent normalised version (bottom) that forms a partition of unity. The dashed lines represent the sum over all the fuzzy sets on this axis.

would use the minimum number of fuzzy terms (*small*, *large*, etc.) to adequately represent an expert's knowledge, and the associated membership functions should have the minimum amount of overlap in order to distinguish the role played by each rule in the overall system's output. No one technique can be used to design a complete natural language system, although methodologies like fuzzy logic can play their part. The most successful fuzzy systems only use a very restricted and well-defined subset of natural language as this makes the systems transparent.

An important point to make about fuzzy logic is that its richer representation is only necessary if the application demands it, i.e. the system's designer must understand the difference between *necessary* and *sufficient* precision. For some applications outside the control world (fuzzy controllers are the most popular application of this technology), a fuzzy set may produce an unnecessarily precise representation of the input measurements and a similar conventional knowledge-based system could be designed with identical input-output behaviour. This generally occurs when the system is designed to produce a symbolic rather than a numeric output, where the fuzzy system must decide in which discrete state the output lies. Consider the example shown in figure 12, where a fuzzy and a conventional set are used to

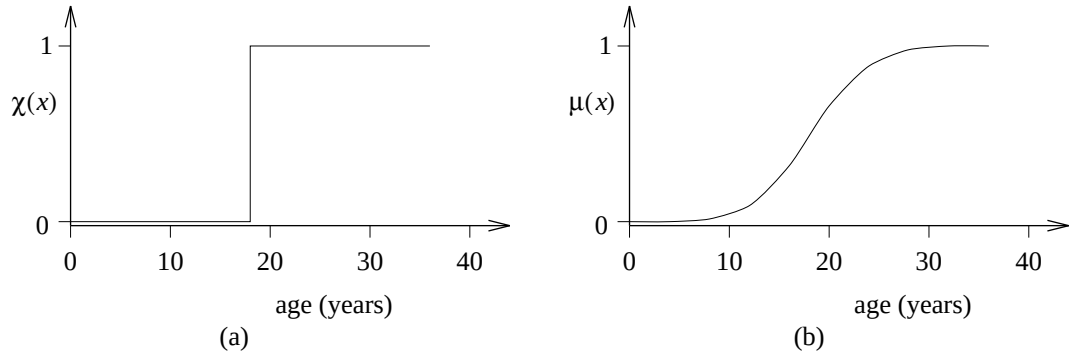


Figure 12: A conventional (a) and a fuzzy (b) representation of the concept of an adult.

represent the term *adult* which is assumed to only depend on a person's age. The task is to

work out whether or not (a crisp decision) a person is an adult. For this decision process, a typical fuzzy rule would be of the form:

IF $(\mu_{adult}(x) > \theta)$ THEN (adult)

and the threshold value θ would determine how much a person had to be considered an adult, before they were officially classified as being one. As the sigmoidal fuzzy membership function is invertible, this threshold decision can be made on the raw measurement or else the fuzzy membership value and the two processes are equivalent.

Fuzzy sets and rules propagate a richer representation (compared to a binary set) throughout the entire system but this is only useful *when* the application requires it. Perhaps the concept of an adult is defined to be different ages for different responsibilities (driving a car, getting married, drinking alcohol), and a fuzzy set would represent this by having different thresholds (although it could be argued that an equivalent result could be obtained using their ages). However, modelling a binary or crisp set with a truly fuzzy membership function would be as inappropriate as representing vague concepts with boolean sets.

2.5 Discrete Fuzzy Sets

So far, this description has concentrated on continuous fuzzy sets. The universe of discourse has been assumed to be a continuous real-valued interval and a well-defined mapping is used to represent each fuzzy membership function. *Discrete* fuzzy sets and systems are based on a universe of discourse that consists of a discrete number of states, and the membership function is represented by assigning a membership value to each state. For example, consider the fuzzy set *fast*, defined on the universe of discourse *cars*; the cars shown in table 1 are members of this set with varying degrees. A subjective, context dependent membership function value is

car	membership value
Ferrari 355	1.0
Reliant Robin	0.0
Ford Escort 1.3L	0.2
Ford Escort XR3i	0.8

Table 1: A discrete fuzzy set representing *fast* cars.

assigned to each car type, as shown in figure 13, where the car-makes have been ordered to represent their degree of fastness.

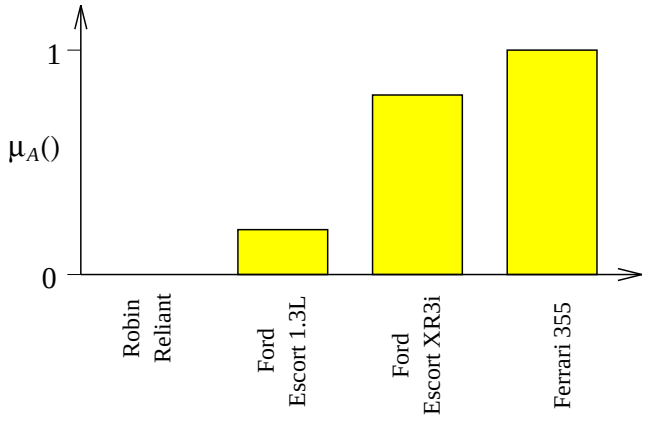


Figure 13: An ordered histogram representing the discrete fuzzy set of *fast* cars.

It is worthwhile making the point that whether to use continuous or discrete membership functions, depends very much on the information provided to the system. For instance, instead of designing a discrete fuzzy set of fast cars, it could be possible to measure the car's top speed (a continuous, real-valued measurement) and to construct a continuous fuzzy set whose input was this signal. However, it could be argued that the top speed of a car is not the only indicator of how someone assesses the car's speed. It is based rather on your impression of the car's make (a Ferrari is always fast) as well as information about its top speed. The former information source would not be modelled by the continuous fuzzy set. Continuous fuzzy systems are based only on a finite number of measurable signals but produce a continuous output. Discrete systems can potentially take an infinite number of information sources into account by receiving an individual's fuzzy classification value as an input (or equivalently the linguistic label which accesses this key). However, this is a subjective, context dependent and unpredictable process and so which representation is best is very much application dependent.

In control engineering, it has been the norm to work with discrete fuzzy sets and systems even though the inputs and outputs are generally real-valued. Attempts have even been made to produce a continuous output from a discrete fuzzy system by linearly interpolating between the rules closest to the input value [33]. However, while this approach achieves its desired objective, it is usual nowadays to work completely with continuous fuzzy systems and it is this approach that will be taken here. Discrete fuzzy membership functions should only be used when the input domain is a set of discrete elements. The relationship between the continuous and discrete approach is described in considerable detail in chapter 10 of [5].

2.6 Fuzzy Set Theory and Probability Theory

Despite the fact the fuzzy set theory was introduced to model linguistic vagueness and probability distributions can be thought of as modelling the uncertainty associated with a particular measurement or process, the differences and the relationships between fuzzy set theory and probability theory have been vigorously debated for as long as fuzzy logic has been in existence [24]. Both are important subject areas as they underpin the vast amounts of work currently being done in the AI area of reasoning under uncertainty.

2.6.1 The Meaning of Fuzzy Membership Functions

One of the roots of this debate is the often asked question by fuzzy designers:

> *How do I obtain my fuzzy membership function?*

This seemingly innocent question has led to considerable in-depth research into the meaning behind modelling a simple linguistic statement such as *x is small*. Consider trying to implement an expert's algorithm which is composed of such fuzzy terms, and trying to understand exactly what is meant by *small*. One possible interpretation is that the membership value is given by:

> $\mu_A(x)$ *is given by the percentage of experts polled who would consider* x *to be a full member of the (fuzzy) set* A.

Here, however, the uncertainty that is being modelled is in the average person's perception of where the crisp concept threshold lies, *not* what is appropriate for the particular problem domain. Also a human's natural tendency to round-off seemingly unimportant details could easily produce biased answers.

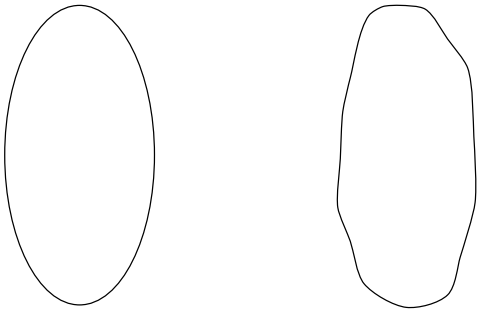


Figure 14: A crisp ellipse and a "fuzzy" ellipse.

Subjective probability is perhaps the most similar concept to fuzzy logic as it measures the uncertainty of an individual's uncertainty about an event or object. For instance, consider classifying the two geometrical figures shown in figure 14. Kosko [20] argued that

> *Does it make more sense to say that the oval is* probably *an ellipse, or that it is a fuzzy ellipse?*

The one on the left is a perfect ellipse, whereas the one on the right may be regarded by some as an *imperfect* ellipse, or some may state that this is not an ellipse. The difference lies in whether or not the designer is trying to model the uncertainty in the drawing process. An ellipse has a precise mathematical definition, and when a geometrical deviates from this ideal, it is up to the observer to determine his belief in that shape being an ellipse. Sources of "noise" could include where the author drew the figure (in a drawing office or on a rolling ferry) and the classification task at hand (extracting ellipses from noisy, pixel-based images or scanning in figures for a mathematical textbook) as well as many other factors. Humans make implicit assumptions about these sources of uncertainty, and assign a subjective measure to the figure. For this case Id argue that the class *ellipse* has a precise mathematical definition and any deviation from this must be assessed in terms of subjective probability.

Whether or not fuzzy logic is regarded as a subset of subjective probability theory (or indeed vice-versa as has been proposed by Kosko [20]), the fuzzy approach has proved useful in many applications, hence it is sufficient for many problems.

2.6.2 Trend Information

Having made these statements about fuzzy sets and probability distributions, it is worth reflecting on the reason why fuzzy logic is used in engineering. Often experts use fuzzy concepts to explain their actions as well as using inherent, domain-specific *trend* information. For example, consider the following three rules:

$$\begin{aligned} & \text{IF } (x \textit{ is small}) \text{ THEN } (o \textit{ is small}) \\ \text{OR} \quad & \text{IF } (x \textit{ is medium}) \text{ THEN } (o \textit{ is medium}) \\ \text{OR} \quad & \text{IF } (x \textit{ is large}) \text{ THEN } (o \textit{ is large}) \end{aligned}$$

which any control engineer would implement as a linear mapping as shown in figure 15. However, modelling each term separately utilising trapezoidal or triangular-shaped membership functions generates different outputs, as shown in figure 15. An expert has an inherent idea about the form of system's output or decision process and this *must* be reflected in the shape of the fuzzy membership functions. Here, there is a potential conflict between system modelling and linguistic representation, but for a fuzzy system to be successful, this type of domain specific knowledge should be encoded in the design of the membership functions.

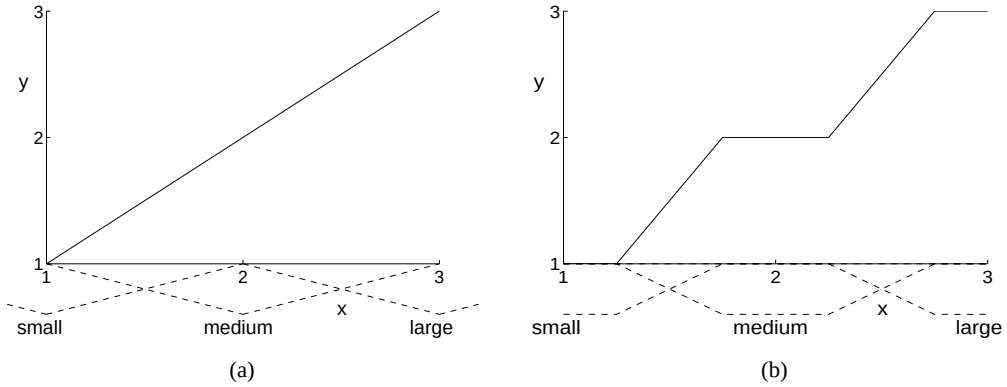


Figure 15: Two different system outputs, (a) linear and (b) piecewise linear/constant, which depend on the form of the fuzzy input membership functions.

2.6.3 Partitions of and Summing to Unity

One of the fundamental axioms of probability theory is that the sum of the distributions should equal one. The events that can happen form a *closed* world, as every possible output is known and a probability can be assigned to each one. Fuzzy logic makes no such restriction as the sum over the set membership values can take any positive value, although it was argued in section 2.3.3 that the fuzzy variables should form a partition of unity, a property that is preserved using algebraic fuzzy operators. The reason for this is that in many fuzzy systems, the fuzzy output set must be defuzzified to produce a real-valued signal. This defuzzification generally imposes an implicit partition of unity on the fuzzy system. Therefore although fuzzy systems do not explicitly require the fuzzy variables to form a partition of unity, it aids the designer if they do and the defuzzification operator generally implicitly imposes such a condition on the overall system.

This normalisation operation can be visualised graphically, using the *fuzzy hypercube* concept. In a fuzzy hypercube, each axis corresponds to the membership of a particular fuzzy set, so each axis is defined on the interval $[0, 1]$ and a fuzzy system with n sets would generate a fuzzy hypercube of dimension $[0, 1]^n$. In figure 16, a system with two ($n = 2$) fuzzy sets is illustrated, and each input can be represented on this diagram by plotting the point which corresponds to the memberships $\mu_{A^1}(x)$ and $\mu_{A^2}(x)$ For a fuzzy variable to form

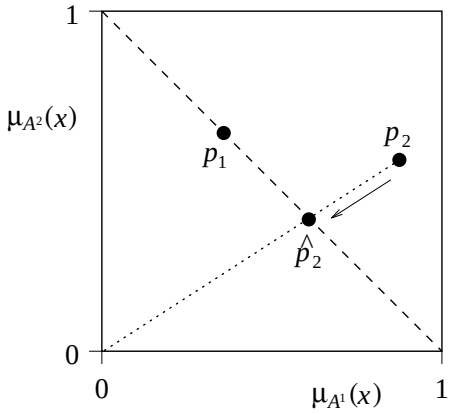


Figure 16: A fuzzy hypercube (square) for two sets A^1 and A^2. The dashed line represents the set of points that sum to unity on which p_1 lies and p_2 is mapped onto this line when the fuzzy sets are required to form a partition of unity.

a partition of unity amounts to requiring that each point should lie on the diagonal line of the fuzzy hypercube generated by:

$$\mu_{A^1}(x) + \mu_{A^2}(x) \equiv 1 \tag{20}$$

When a fuzzy variable is normalised to form a partition of unity, this corresponds to *sliding* the point down the (dotted) line it makes with the origin until it reaches the diagonal (dashed) line (see figure 16). Whenever a normalising defuzzification operator is used, this implicitly imposes a partition of unity on the fuzzy variables. Hence even though a fuzzy system does not require the membership to sum to unity, a normalising operation is generally performed on all the membership functions.

In summary, subjective probability and fuzzy logic are quite similar in the type of uncertainty each technique tries to model. However, because fuzzy relations focus directly on the input, output mapping which is arguably more natural for domain experts (as opposed to deriving conditional probabilities), and because of the flexibility in deriving an appropriate membership function shape, it has proved useful in many engineering applications.

Fuzzy logic is a sound theory for generalising conventional boolean concepts to the vague, real-world, as has been shown in numerous applications.

3 Fuzzy Operators

Fuzzy sets and membership functions are the reason why fuzzy logic was introduced; since they provide a means of representing the concept of vague membership of a set. However, this ability to map data to fuzzy set memberships is not useful in itself as we also require a set of operators for combining this information and making inferences about its state. Fuzzy logical operators provide this flexibility and this section describes some of the most common.

Fuzzy operators are generalisations of the common boolean logical operators such as AND, OR, NOT etc., and for the binary characteristic functions, these are well-defined in terms of truth tables. Perhaps the most common implementation of these functions has been using the following truncation operators:

$$\mu_{A \text{ AND } B}(x) = \min\{\mu_A(x), \mu_B(x)\} \tag{21}$$

$$\mu_{A \text{ OR } B}(x) = \max\{\mu_A(x), \mu_B(x)\} \tag{22}$$

$$\mu_{\text{NOT } A}(x) = 1 - \mu_A(x) \tag{23}$$

where all of these operators map the unit interval to the unit interval, so their output can be interpreted as a membership value of a "new" compound fuzzy set.

3.1 Boolean Operators

Conventional boolean logical operators such as AND, OR, NOT are well-defined in terms of truth tables, as illustrated in figure 17. The logical operators are classed as either:

unary a function with only one argument, i.e. NOT, which maps the binary set $\{0, 1\}$ onto itself (this can only be the identity mapping or its complement), and

binary a function with two arguments, i.e. AND, OR, which is defined on $\{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$.

where (obviously) a logical function with more than two arguments, can be written as a composition of several binary functions.

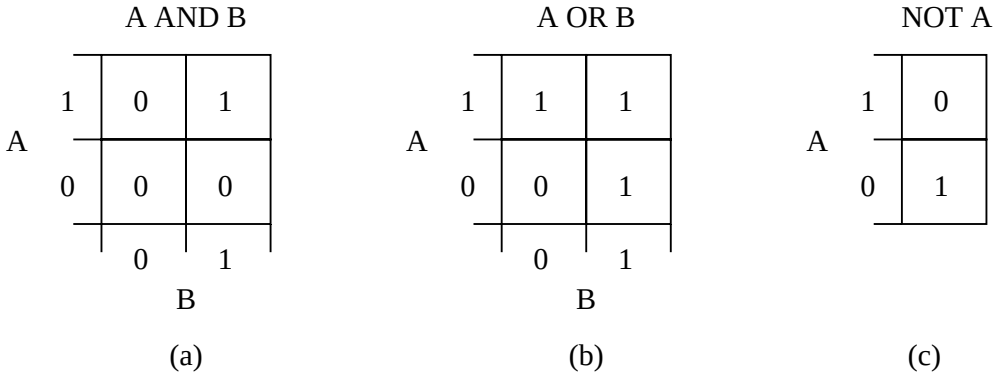


Figure 17: Truth tables for the logical AND (a), OR (b) and NOT (c) operators.

These logical operators correspond to the union, intersection and complement functions in set theory. The ANDing of two sets refers to the intersection, ORing is equivalent to finding which members lie in the union of the sets and NOT is simply the complement of the original set. This is illustrated in figure 18, where the corresponding Venn diagrams are shown.

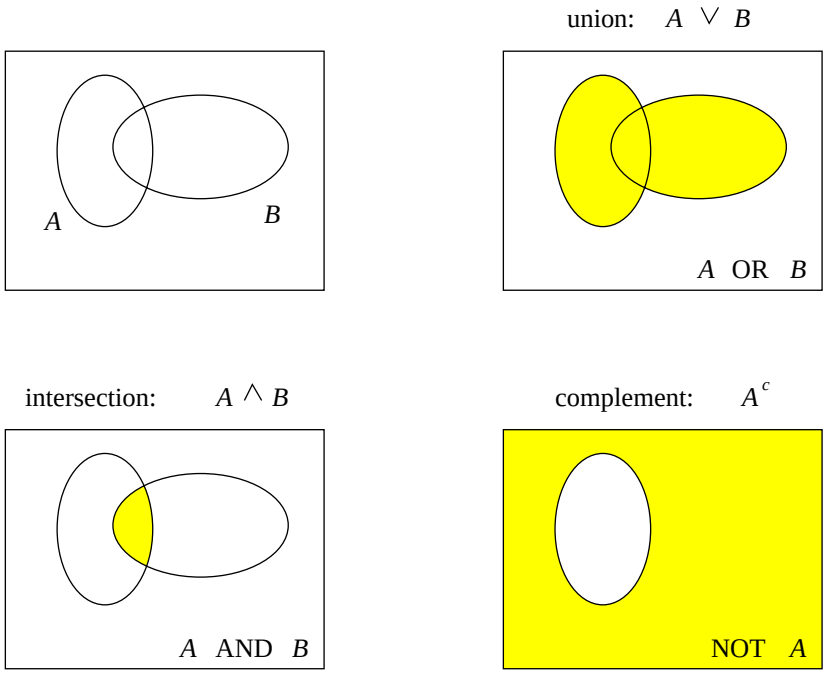


Figure 18: The Venn diagrams that correspond to intersection (AND), union (OR) and complement (NOT).

Boolean operators have as inputs the value of the appropriate characteristic function, and produce an output that represents the output of the compound characteristic function. Fuzzy logical operators perform a similar mapping, generating new membership functions from their input membership functions.

Fuzzy set theory allows membership values of between 0 and 1, and hence new fuzzy operators must be found for intersection, union, negation, implication etc., as such operations cannot be stored in a tabular form. Zadeh originally used the min and max operators as they are simple to implement, are equivalent to the Boolean operators for binary arguments and always map the unit interval to the unit interval. These *truncation* operators were used almost

exclusively during the seventies and most of the eighties, and it is only recently that other operators have been seriously considered, [5, 9, 34]. These alternative *algebraic* operators form the basis for neurofuzzy systems which we shall come across in section 5.

3.2 Law of the Excluded Middle

As long ago as 1923, Bertrand Russell [31] noted that:

> *The law of the excluded middle (*A AND NOT A*) is true when precise symbols are employed but it is not true when symbols are vague, as, in fact, all symbols are.*

The A AND NOT A excluded middle concept is one of the foundations of classical logic, as it asserts that:

$$\begin{aligned} A \text{ AND NOT } A &\equiv 0 \\ A \text{ OR NOT } A &\equiv 1 \end{aligned} \tag{24}$$

and this is at the heart of many mathematical proofs by contradiction, as the law of the excluded middle asserts that a contradiction of the negated concept implies that the original statement was true. For instance, the following proof by contradiction shows that there exists an infinite number of prime numbers.

Example 3.1 (Proof by contradiction) *Assume that there is only a finite number of primes* $\{p_i\}_{i=1}^{n}$*, and construct the number:*

$$p = \left(\prod_{i=1}^{n} p_i \right) + 1$$

Each p_i *cannot be a factor of* p *and it also satisfies the relationship* $p_i < p$*. Hence either* p *is a prime number or else there exists a prime number* q *such that* $p_i < q < p$ *for all* i*. In either case, the original assertion is false and so there must exist an infinite number of prime numbers.*

However, suppose that the degree of membership of the set A now lies in the interval $[0, 1]$ rather than only at its edges. The membership of the set NOT A would also be vague and the law of the excluded middle could not hold, as an element can be a partial member of a set and its complement at the same time. As an example of this consider the fuzzy set which represents the concept *adult*, shown in figure 12. Figure 19 draws a representation of the sets NOT A and A AND NOT A. Someone who is around 18 years of age can be considered

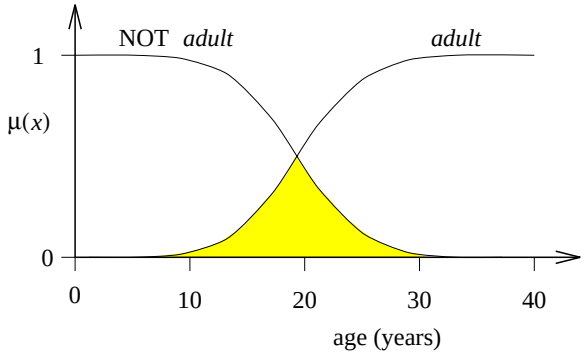


Figure 19: The fuzzy set (*adult* AND NOT *adult*) shown as the shaded region.

both an adult and not an adult at the same time. This violation of the law of the excluded middle is typical of real-world classes where someone can be *old* AND NOT *old*, or *rich* AND NOT *rich*, as this specifies both a region and a membership function which gives the overlap between A AND NOT A.

A crisp set satisfies the law of the excluded middle and has no overlap between the sets A and NOT A, hence Kosko [22] proposed it as a basis for the measure of the fuzziness of a set and termed it *fuzzy entropy*.

3.2.1 Fuzzy Entropy

Fuzzy entropy is the measure of a particular set's fuzziness, and is defined by the formula:

$$E(A) = \frac{c(A \text{ AND NOT } A)}{c(A \text{ OR NOT } A)} \tag{25}$$

where c refers to a count (addition or integration) over all the corresponding membership values. For a crisp set, the numerator is always zero and the denominator is always unity, hence the fuzzy entropy of a crisp set is zero.

Fuzzy entropy is a measure of a sets own fuzziness, although its influence on the overall system is determined by its interaction with neighbouring sets. The standard fuzzy entropy measure tells you nothing about this, although using the normalised fuzzy membership functions defined in equation 19, results in the complement of a set being the union (algebraic sum) of all the remaining sets. Therefore, the fuzzy entropy in this case would be a measure of the fuzzy set's interaction with neighbouring membership functions.

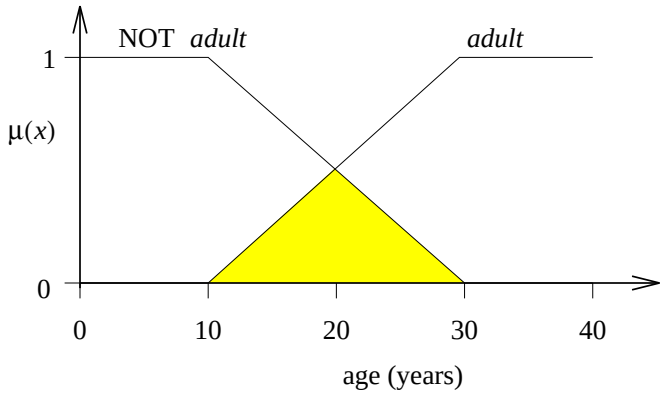


Figure 20: The shaded shape represents the fuzzy set A AND NOT A, whose relative area determines the fuzzy entropy of a set.

This is illustrated in figure 20 where its is shown how the entropy of a fuzzy set can be calculated, using the sum operator to represent OR (as the fuzzy membership variable forms a partition of unity) and max operator to represent AND. The fuzzy entropy measure is therefore the ratio between the area between the intersection of the membership function and its complement, and the total size of the universe of discourse. In this example, $X = [0, 40]$ hence:

$$\begin{aligned} c(A \text{ AND NOT } A) &= 5 \\ c(A \text{ OR NOT } A) &= 40 \end{aligned}$$

and the fuzzy entropy of A is given by:

$$E(A) = 0.125$$

The concept of fuzzy entropy is sometimes useful as it indicates the relative amount of overlap, but it is a fairly course measure.

3.3 Fuzzy Rule Bases

Often, experts will articulate their knowledge in terms of simple IF - THEN production rules which map an input state directly to an output. The vague linguistic premises and conculsions such as:

the error is postive small.

or

the system's response is almost zero.

can be represented using fuzzy sets and combined using fuzzy operators and for much of the remainder of this course, we'll be studying the affect of different implementation methods.

A *fuzzy algorithm* is usually composed of fuzzy production rules of the form:

$$\begin{array}{lll} r_{1,1}: & \text{IF } (\mathbf{x} \textit{ is } \mathbf{A}^1) \text{ THEN } (o \textit{ is } B^1) & 0.3 \\ r_{1,2}: \text{ OR} & \text{IF } (\mathbf{x} \textit{ is } \mathbf{A}^1) \text{ THEN } (o \textit{ is } B^2) & 0.7 \\ \vdots & \vdots & \\ r_{i,j}: \text{ OR} & \text{IF } (\mathbf{x} \textit{ is } \mathbf{A}^i) \text{ THEN } (o \textit{ is } B^j) & 1.0 \\ \vdots & \vdots & \\ r_{p,q}: \text{ OR} & \text{IF } (\mathbf{x} \textit{ is } \mathbf{A}^p) \text{ THEN } (o \textit{ is } B^q) & 0.0 \end{array} \tag{26}$$

where $r_{i,j}$ rule is the ij^{th} fuzzy production rule which relates the i^{th} input fuzzy set, $\mathbf{A}^i$, to the j^{th} output fuzzy set, B^j. In more detail, each fuzzy production rule has a structure of the form:

$$\text{IF } (x_1 \textit{ is } A_1^i \text{ AND } \cdots \text{ AND } x_n \textit{ is } A_n^i) \text{ THEN } (y \textit{ is } B^j) \qquad c_{ij} \tag{27}$$

or linguistically as:

$$\text{IF (antecedent) THEN (consequent)} \qquad c_{ij} \tag{28}$$

The degree or confidence with which the input fuzzy set $\mathbf{A}^i$ (which is composed of the fuzzy intersection (AND) of several univariate fuzzy sets) is related to the output fuzzy set B^j is given by a *rule confidence* $c_{ij} \in [0, 1]$. When c_{ij} is zero, the rule is never active and hence does not contribute to the system's output. Otherwise the rule partially fires whenever its antecedent is activated to a degree greater than zero. Therefore, the rule base is characterised by the set of rule confidence $\{c_{ij}\}$ $(i = 1, 2, \ldots, p,\ j = 1, 2, \ldots, q)$, and these can naturally be stored in a rule confidence matrix $\mathbf{C}$ whose ij^{th} element is c_{ij}. A zero entry in the rule confidence matrix means that the corresponding rule does not influence the system in any way. Once the fuzzy membership functions have been defined, the rule confidences encapsulate the expert's knowledge about a particular process and they also form a convenient set of parameters to train.

The vast majority of fuzzy systems successfully deployed are *flat* in the sense that the rules directly relate the system's input to its output. There are no intermediate or *hidden* states. *Deep* fuzzy systems have the potential to represent the desired input, output mapping using a lot fewer rules, see section 6.5.1, although is is quite difficult to determine an appropriate representation if its not natural for that particular application.

Before delving deeper into how these fuzzy rule bases are represented in a computer, it is worth discussing what a fuzzy algorithm actually represents. Originally, it was proposed as a technique for modelling the way humans think and reason, although this rather grand idea has now been replaced with the thought that a fuzzy algorithm is simply the linguistic *interface* between humans and computers. Because humans explain much of their actions using vague, linguistic statements, it makes sense to develop techniques which are capable of implementing this knowledge in a precise, but appropriate manner. Indeed, the authors take the view that:

> *irrespective of how humans think and reason, they* explain *their actions using vague, linguistic statements. Fuzzy logic is one technique for representing this knowledge on a computer.*

Hence, we take a fairly pragmatic view about the usefulness of fuzzy logic, in that a fuzzy approach will only be successful when:

1. there exists sufficient linguistic expert knowledge to completely characterise the solution to the problem, and
2. fuzzy logic is an appropriate mechanism for representing this knowledge.

3.3.1 Fuzzy Rule Confidences

The rule confidences depend neither on the shape or form of the fuzzy sets, nor on the fuzzy logical operators, both of which are stored *separately* in the knowledge base. Discrete fuzzy systems, which have been widely used in self-organising controllers, construct a relational matrix which completely characterises the knowledge base as it implicitly contains information about the fuzzy set shapes, logical operators and rule confidences [13, 29]. Storing knowledge in a distributed fashion as has been described is preferable as it makes it easier to understand how differing implementation methods will affect the system's output.

Associated with each multivariate fuzzy input set is a rule confidence vector $\mathbf{c}_i$ which represents the estimated output of the system for that particular input set. These rule confidence vectors are generally normalised (sum to unity) as this implies that there is total knowledge about the system's output for that particular input set. These parameters can easily be updated when the knowledge in the rule base is changed. In a lot of adaptive fuzzy systems, the fuzzy output membership functions are altered by shifting their centres which amounts to redefining the designer's subjective interpretation of a linguistic statement. It could therefore be argued that these adaptive fuzzy systems cannot be validated after training because the form of the fuzzy sets is not consistent with their original definition. However, when rule confidences are used and stored separately from the fuzzy sets, it is possible to adapt the strength with which a rule fires and still retain its original fuzzy, linguistic interpretation.

3.3.2 Terminology

Completeness and *inconsistency* of rule bases are two concepts that have well-defined meanings in conventional expert systems, which must be generalised to be used in fuzzy systems.

Definition 3.1 (Rule Base Completeness) *A rule base is said to be* complete *if for each* $\mathbf{x} \in \mathbf{X}$*, there exists a* o *such that:*

$$\mu_R(\mathbf{x}, o) > 0 \tag{29}$$

where $\mu_R(\mathbf{x}, o)$ is the membership function for the complete rule base defined on $\mathbf{X} \times O$, which is known as the *relational surface*. It is obtained from the union of all of the individual rules, as described in section 3.6.1.

In general, any rule base that uses membership functions with a non-compact support will be complete, as each rule's membership function will be non-zero over the whole input/output space. However, this would provide the designer will little, or no, information about the rule base's coverage, and a possibly more useful measure would be described as α-completeness.

Definition 3.2 (Rule Base α-Completeness) *A rule base is said to be* α-complete *if the α-cut relational surface is complete.*

Note that this definition takes into account both the membership functions used and the value of the rule confidences. When the rules are binary ($c_{ij} \in [0, 1]$), the definitions of completeness only take into account whether there exists a rule antecedent membership function which covers that part of the input space and is equivalent to fuzzy variable completeness (see section 2.2.1).

Definition 3.3 (Rule Base Inconsistency) *A set of fuzzy rules are said to be* inconsistent *if two rules that have the same linguistic fuzzy antecedent map to two* non-overlapping *fuzzy output sets.*

There are several commonly used definitions of rule base consistency and inconsistency, but this one is especially important for the relationship between the fuzzy and neurofuzzy systems which will be described in this report. When two rules with the same antecedent map to different, overlapping output sets, this can be interpreted as meaning that the overall output associated that particular input set should lie somewhere between the output sets' centres, in the area of overlap. When the output sets are non-overlapping, then either the fuzzy output variable has an inappropriate representation or else the rule base is inconsistent.

3.4 Fuzzy Intersection: AND

The fuzzy *intersection* of two sets A and B refers to a linguistic statement of the form:

$$x \text{ is } A \text{ AND } y \text{ is } B$$

where x and y could potentially refer to the same variable. A *new* fuzzy membership function is generated by this operation defined on $X \times Y$ space, and is denoted by $\mu_{A \cap B}(x, y)$, where the fuzzy $\cap$ notation is an obvious generalisation of the binary $\wedge$ symbol. For binary arguments (crisp sets), these operators are well-defined and can be tabulated in a truth-table, but there are many possible generalisations for fuzzy logic. The family of potential operators is known as the set of triangular norms, or *T-norms*, and the new membership function is generated by:

$$\mu_{A \cap B}(x, y) = \mu_A(x) \hat{*} \mu_B(y) \tag{30}$$

where $\hat{*}$ is the T-norm operator. For ease of notation in the following definition, let $a, b, c, d \in [0, 1]$ denote the value of the fuzzy membership functions.

Definition 3.4 (T-norm) *The set of triangular norms, or T-norms, is the class of functions which obey the following relationships:*

1. $a \hat{*} b = b \hat{*} a$
2. $(a \hat{*} b) \hat{*} c = a \hat{*} (b \hat{*} c)$

3. *if* $a \leq c$ *and* $b \leq d$ *then* $a \hat{*} b \leq c \hat{*} d$

4. $a \hat{*} 1 = a$

In addition, a T-norm is said to be *Archimedean* when:

$$a \hat{*} a < a \qquad \forall a \in (0, 1) \tag{31}$$

There are many possible operators which satisfy these conditions, but the two most commonly used are the product and min functions:

$$\mu_{A \cap B}(x, y) = \mu_A(x) * \mu_B(y) \tag{32}$$

$$\mu_{A \cap B}(x, y) = \min \{\mu_A(x), \mu_B(y)\} \tag{33}$$

for which the former is an Archimedean T-norm. Historically, the min operator was used since it was emphasised by Zadeh when he started writing about fuzzy logic, but more recently the algebraic product operator has been shown to perform better in many situations, although the correct one to use is very much situation dependent. It can be shown that for any T-norm:

$$\mu_A(x) \hat{*} \mu_B(y) \leq \min \{\mu_A(x), \mu_B(y)\} \tag{34}$$

therefore the min operator forms an upper bound on the space of fuzzy intersection operators.

It is useful to visualise the intersection operator graphically, and in figure 21, a 2-dimensional fuzzy membership function formed from the product of two triangular (B-splines of order 2) membership functions is shown. Obviously, the shape of the multivariate fuzzy

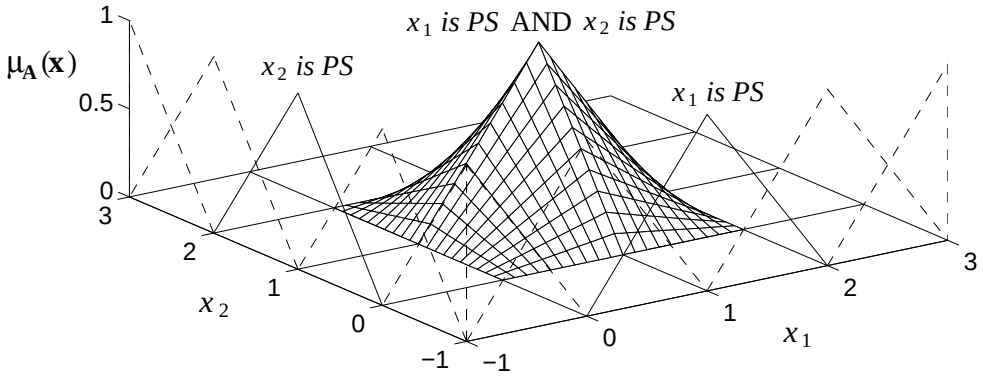


Figure 21: A two-dimensional fuzzy membership function formed from the intersection (product operator) of two triangular, univariate fuzzy membership functions.

membership function depends on both the shapes of the univariate membership functions and the operator used to represent the T-norm. The multivariate membership functions formed using the product operator retain more information than when the min operator is used to implement the fuzzy AND because the latter scheme only retains one piece of information whereas the product operator depends on both pieces. Using the product operator also allows error information to be back propagated through the network as the first derivative is well-defined. It also generally gives a smoother output surface (as will be demonstrated later), as when univariate B-spline and Gaussian fuzzy membership functions are used to represent each linguistic statement, the multivariate membership function is simply a multi-dimensional B-spline or Gaussian basis function, which is illustrated in figure 22.

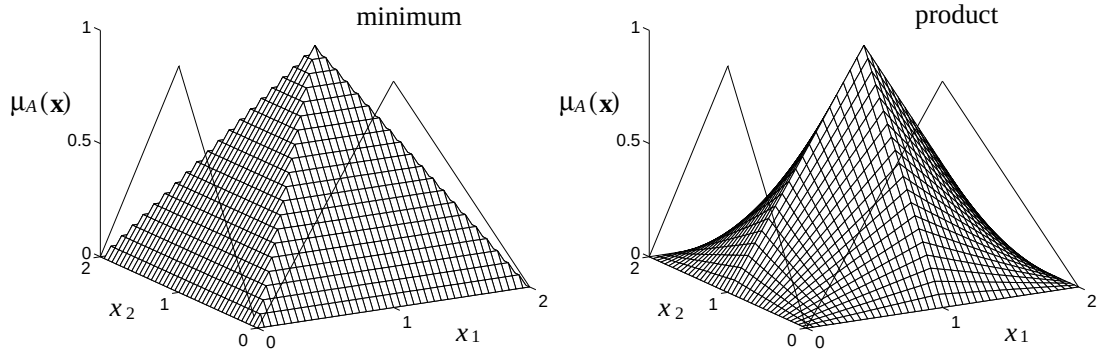


Figure 22: A comparison between the minimum and product fuzzy intersection operators.

In a fuzzy algorithm, the antecedent of a fuzzy production rule is formed from the fuzzy intersection of n univariate fuzzy sets:

$$x_1 \textit{ is } A_1^i \text{ AND } \cdots \text{ AND } x_n \textit{ is } A_n^i$$

which produces a new *multivariate* membership function $\mu_{A_1^i \cap \cdots \cap A_n^i}(x_1, \ldots, x_n)$ or $\mu_{A^i}(\mathbf{x})$ defined on the original n-dimensional input space and whose output is given by:

$$\mu_{A^i}(\mathbf{x}) = \widehat{\prod} \left(\mu_{A_1^i}(x_1), \ldots, \mu_{A_n^i}(x_n) \right)$$

where $\widehat{\prod}$ is the multivariate T-norm operator.

3.4.1 Fuzzy Logical Connectives and Probability Theory

Fuzzy sets and probability density functions may initally appear to have a similar shape and functionality but their interpretations are very different. Similar comments can be made about fuzzy connectives and probability operators, as the probability that two events x and y occur can be calculated from:

$$\Pr(x\, y) = \Pr(x|y) * \Pr(y) \tag{35}$$

where $\Pr(x|y)$ is the probability that x will occur given that y has occured. When the two events are totally dependent:

$$\Pr(x\, y) = 1 * \Pr(y) = \Pr(y) \tag{36}$$

as this corresponds to a linguistic statement such as:

$$\textit{John is tall} \text{ AND } \textit{John is tall}$$

When the events are completely independent:

$$\Pr(x\, y) = \Pr(x) * \Pr(y) \tag{37}$$

this corresponds to a linguistic statement such as:

$$\textit{John is tall} \text{ AND } \textit{Mary is short}$$

Fuzzy logic combines individual membership function values to obtain the set membership of the 2-dimensional set, and so there does not exist any concepts of terms like $\Pr(x|y)$.

The two fuzzy operators that have been discussed so far, min and product, evaluate to the same value as the probability calculation when the events are totally dependent and independent, respectively. However, this does *not* imply that under some circumstances fuzzy and probability theory are equivalent, rather it is used to illustrate that the operators which combine different uncertainty or vague measures are similar. In fuzzy systems, the type of logical operators used are chosen independently from the statistics of the input signal and as such cannot be related to probability theory

It can be argued that in most situations the product operator is more natural and gives the system a smoother output. In [9], they give an example of a prisoner breaking out through two windows, where the ease with which a prisoner can get through each window is 0.3 and 0.1. It is argued that the ease with which a prisoner can escape should be given by $\min\{0.3, 0.1\} = 0.1$, although they do note that the prisoner will become somewhat tired getting through the first window. Taking this argument to its extremes, you could imagine a situation where there were twenty windows that were 0.3 easy to get through and one window which was 0.1 easy to get through. Common sense indicates that the ease with which a prisoner will escape would be quite a bit less than 0.1, but the min operator does not reflect this. Using the product operator means that all of the properties of the fuzzy variables, such as partitions of unity, will be retained by the set of multivariate fuzzy membership functions.

3.4.2 Multivariate Fuzzy Input Set Distribution

When all possible fuzzy intersections are taken of n sets of fuzzy membership functions, this implicitly generates an n-dimensional lattice in the original input space on which the new, multivariate fuzzy membership functions are defined. As illustrated in figure 23, when the fuzzy intersection is taken of every possible combination of univariate fuzzy input sets, the

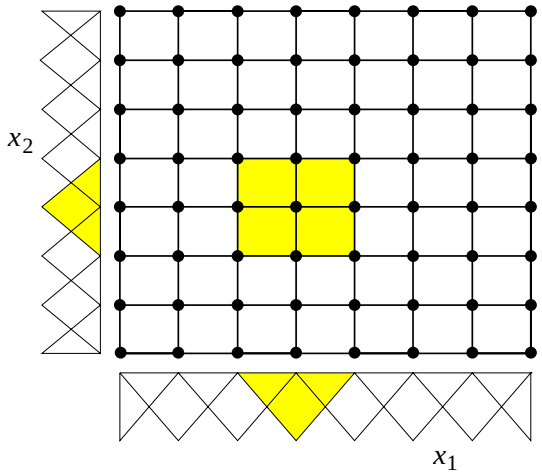


Figure 23: A complete set of 2-dimensional fuzzy membership functions generated by two sets of triangular, univariate fuzzy sets. The bold circles denote their centres and the shaded area illustrates how two univariate fuzzy sets are combined using the intersection operator.

number of multivariate fuzzy membership functions is an *exponential* function of the number of input variables (note that the y axis is scaled logarithmically).

3.4.3 Curse of Dimensionality

The **Curse of Dimensionality** was a phrase coined by Bellmann in 1961 [2] and it refers to the exponential increase in resources required by a system when the input space dimension

increases. For a complete, lattice-based fuzzy system, the number of combinations of the linguistic input terms is:

$$p = \prod_{i=1}^{n} p_i \tag{38}$$

where the i^{th} fuzzy variable is composed of p_i fuzzy sets (univariate basis functions). For a typical fuzzy system that has 7 sets on each axis, p is plotted against n in figure 24. This exponential increase in the number of multivariate fuzzy membership functions has

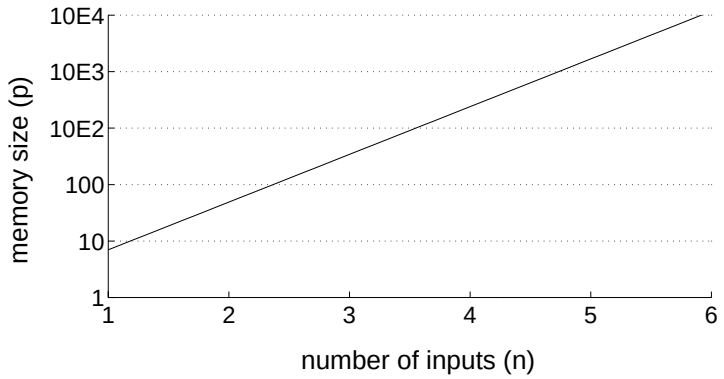


Figure 24: The number of possible combinations of linguistic inputs (p_i^n) plotted against the size of the input space.

implications not only for the size of the fuzzy system, but also for the calculation time and the amount of training data for adaptive fuzzy systems.

A fuzzy system which has triangular membership functions as described above is always *complete*, if and only if each of the fuzzy variables are complete and every combination of linguistic terms is taken. Removing just one of the 2-dimensional fuzzy sets in figure 23 would mean that the rule-base was no longer complete since the membership of every basis function is zero at the centre of the missing set; this is a consequence of the compact support property of the triangular membership functions. Therefore, unless special techniques are used to structure the inputs to a fuzzy network, these systems suffer from the curse of dimensionality, which limits their application to small-dimensional (2 or 3 inputs) engineering problems.

Multi-Layer Perceptron networks do not directly suffer from the curse of dimensionality as the computational cost (size) of the network depends on the number of nodes in the hidden layer (as well as the number of hidden layers). Each node in the hidden layer has an associated weight vector where each element multiplies a corresponding input, so its activation value is given by:

$$i = \sum_{i=0}^{n} w_i x_i \tag{39}$$

Hence including an extra input simply increases the number of parameters in the weight vector by 1. The number of nodes required in the hidden layer is determined by the complexity of the data, and the more hidden nodes in the network, the more complex a mapping can be produced as each hidden layer node effectively splits the input space into two regions, in the same manner as a single Perceptron node. If the desired mapping is very complex, the number of hidden layer nodes may be exponentially dependent on the size of the input space, although a lot of training data will be required.

Many static fuzzy systems implement only a small number of these rules, as it is difficult for an expert to correctly articulate more than about 100 production rules. However, this leaves the possibility that the fuzzy system may no longer be complete, and hence its overall

behaviour is more difficult to verify and validate. We shall return to this problem in section 6 where simple (additive) fuzzy systems are described and extra resources are included, depending on the complexity of the underlying data.

3.4.4 Variable Independence

The concept of *independence* is fundamental in probability theory and it is useful to consider the choice of the fuzzy intersection operator from this perspective. Probability theory says that two events A and B are independent if:

$$\Pr(AB) = \Pr(A)\Pr(B) \tag{40}$$

and the probability of them both occurring is simply equal to the product of the individual probabilities. If however the two events are dependent, the probability of AB occurring is given by:

$$\Pr(AB) = \Pr(A)\Pr(A|B) \tag{41}$$

and when the two events are the same, $\Pr(A|B) = 1$, this simply reduces to a form which is consistent with using the min operator.

In fuzzy logic, the product and min operator have proved to be the most popular choices for representing fuzzy intersection and as can be seen from the preceding discussion this corresponds to the two extremes of two events being totally independent and dependent, respectively. However, except in the simplest of cases, these concepts do not necessarily apply to fuzzy logic. Obviously, when A and B are equivalent, the non-Archimedean min operator should be used:

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} = \mu_A(x)$$

as this corresponds to a linguistic statement such as:

John is Tall AND *John is Tall*

In this case, the univariate fuzzy membership functions can be recovered by projecting the multivariate fuzzy membership function back onto any of the input axes, because intersection is represented using the min operator.

3.5 Fuzzy Implication: IF () THEN ()

Fuzzy implication relationships are used to encapsulate an expert's knowledge about how a vague linguistic input set is related to an output set. This can be represented as:

$$A \to B \tag{42}$$

or linguistically as:

$$B \text{ is true whenever } A \text{ is true} \tag{43}$$

However, its usage in fuzzy systems generally differs from its common definition in standard and multi-valued logics. In standard binary logic, implication is represented as:

$$\begin{aligned} A \to B &= A^c \vee B \\ &= (A \wedge B) \vee A^c \end{aligned}$$

and has a truth table given shown in figure 25. Thus even when A is false, B will be activated as the implication is true.

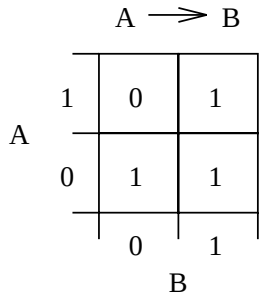


Figure 25: Boolean implication truth table.

In fuzzy systems, implication represents a causal relationship between input and output sets, where the ideas of *local* knowledge representation are particularly important. Rule confidences store the strength of the association between A and B, and it would be unreasonable to adapt every rule confidence in the rule confidence matrix based on a *single* piece of knowledge. In a fuzzy system which has fuzzy variables that form partitions of unity, NOT A refers to every other membership function and even if the data doesn't activate these sets, the rule confidences will be set to non-zero values which depends on the activation of B. This is undesirable as you'd expect that a locally defined rule should only have local influence, and so fuzzy implication is often treated as a generalised intersection operator where the rule confidence is changed if and only if both A and B are non-zero.

Therefore, to represent a relationship (IF *antecedent* THEN *consequent*), let the rule that maps the i^{th} multivariate fuzzy input set A^i to the j^{th} univariate output set B^j with a confidence c_{ij} be labelled by r_{ij}, i.e.:

$$r_{ij}: \qquad \text{IF } (\mathbf{x} \textit{ is } A^i) \text{ THEN } (y \textit{ is } B^j) \qquad c_{ij}$$

Then the degree to which element x is *related* to element y is represented by the $(n+1)$-dimensional membership function $\mu_{r^{ij}}(\mathbf{x}, y)$ defined in the product space $A_1 \times \cdots \times A_n \times B$ by:

$$\mu_{r^{ij}}(\mathbf{x}, y) = \mu_{A^i}(\mathbf{x}) \hat{*} c_{ij} \hat{*} \mu_{B^j}(y) \tag{44}$$

where $\hat{*}$ is the triangular norm usually chosen to be the min or the product operator. The fuzzy set $\mu_{r^{ij}}(\mathbf{x}, y)$ represents the confidence in the output being y given that the input is $\mathbf{x}$ for the ij^{th} fuzzy rule.

There are several other methods of implementing the implication operator and the interested reader is referred to [9] for a good discussion of their merits. However, the one described above is particularly important as it allows a relationship to be made between fuzzy and neural systems.

3.6 Fuzzy Union: OR

The fuzzy *union* of two sets A and B refers to a linguistic statement of the form:

$$x \textit{ is } A \text{ OR } y \textit{ is } B$$

where x and y could potentially refer to the same variable. A *new* fuzzy membership function is generated by this operation defined on $X \times Y$ space, and is denoted by $\mu_{A \cup B}(x, y)$, where the fuzzy $\cup$ mirrors the binary $\vee$ symbol. Once more, fuzzy and binary union are equivalent for binary arguments, but there exist many ways of generalising it in fuzzy logic. This family

of operators is known as triangular co-norms (*S-norms*), and the new membership functions are obtained from:

$$\mu_{A\cup B}(x,y) = \mu_A(x)\widehat{+}\mu_B(y) \tag{45}$$

where $\widehat{+}$ is the binary S-norm operator. Again letting $a, b, c, d \in [0,1]$ denote the value of the fuzzy membership functions, we have the following definition.

Definition 3.5 (S-norm) *The set of triangular co-norms, or S-norms, is the class of functions which obey the following relationships:*

1. $a\widehat{+}b = b\widehat{+}a$
2. $(a\widehat{+}b)\,\widehat{+}c = a\widehat{+}\,(b\widehat{+}c)$
3. *if* $a \leq c$ *and* $b \leq d$ *then* $a\widehat{+}b \leq c\widehat{+}d$
4. $a\widehat{+}0 = a$

Two of the most commonly used operators that satisfy these conditions are the sum and max functions:

$$\mu_{A\cup B}(x,y) = \mu_A(x) + \mu_B(y) \tag{46}$$
$$\mu_{A\cup B}(x,y) = \max\{\mu_A(x), \mu_B(y)\} \tag{47}$$

and if the fuzzy membership functions do not form a partition of unity, the sum operator is sometimes replaced with the bounded sum:

$$\mu_{A\cup B}(x,y) = \mu_A(x) + \mu_B(y) - \mu_A(x) * \mu_B(y) \tag{48}$$

which always has a membership value that lies in the unit interval.

The max operator can be shown to be the most pessimistic S-norm as:

$$\max\{\mu_A(x), \mu_B(y)\} \leq \mu_A(x)\widehat{+}\mu_B(y) \tag{49}$$

Unlike the class of T-norms, when the arguments of an S-norm are unimodal, the resulting membership function is unlikely to retain this property. This is illustrated in figure 26, where the max and sum operators are compared and it can clearly be seem that non-unimodal membership functions may be produced by the union operator.

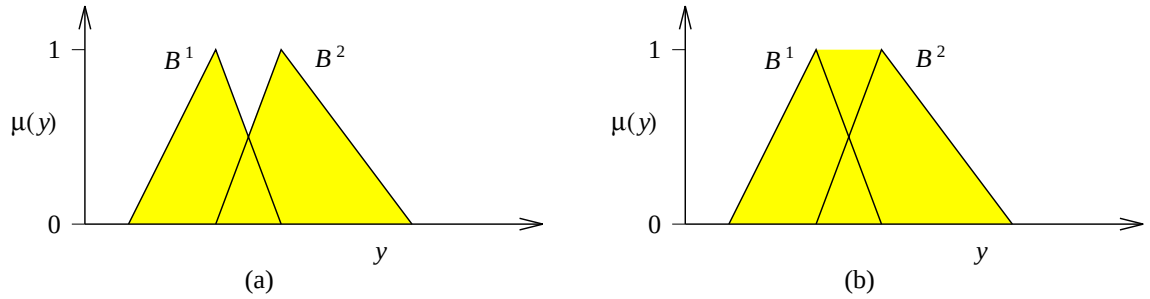


Figure 26: A comparison of the max and sum union operators, where the shaded area represents the membership function $\mu_{B^1\cup B^2}(.)$.

3.6.1 Fuzzy Relational Surfaces

For simple, fuzzy algorithms, the union operator is generally used to connect the outputs of different rules, although it is also informative to consider the union of all the rules defined on both the input and output universes. When p multivariate fuzzy input sets A^i map to q univariate fuzzy output sets B^j, there are pq overlapping $(n+1)$-dimensional membership functions formed using the intersection and implication operators, one for each relation. The pq relations can then be connected to form a fuzzy rule base R by taking the union (OR) of the individual membership functions, and this operation is defined by:

$$\mu_R(\mathbf{x}, o) = \widehat{\sum}_{i,j} \mu_{r^{ij}}(\mathbf{x}, o) \tag{50}$$

where $\widehat{\sum}$ is the multivariable S-norm operator.

The union of all the individual relational membership functions forms a *ridge* or *relational surface* in the input/output space which represents how individual input/output pairs are related and can be used to infer a fuzzy output membership function given a particular input measurement; a process known as fuzzy inferencing. A typical relational surface is shown in figure 27, where four triangular fuzzy sets (B-splines of order 2) are defined on each variable,

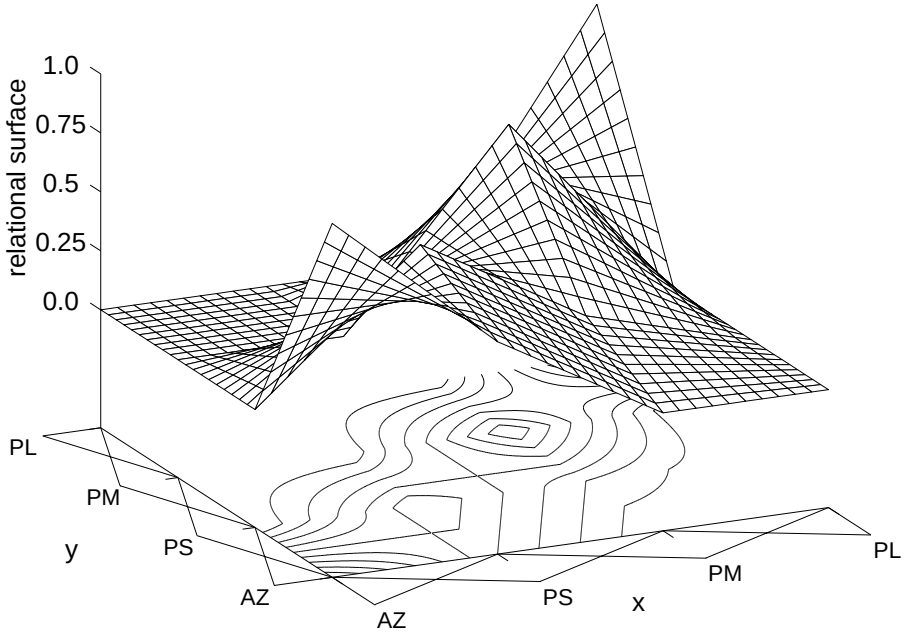


Figure 27: A fuzzy relational surface, $\mu_R(\mathbf{x}, o)$, and associated contour plot for a single input, single output fuzzy system with normalised rule confidence vectors and algebraic fuzzy operators. Each peak corresponds to a fuzzy rule.

the algebraic functions are used to implement the logical operators and the fuzzy algorithm is given by:

$r_{1,1}$		IF (x *is* AZ) THEN (o *is* AZ)	1.0
$r_{2,1}$	OR	IF (x *is* PS) THEN (o *is* AZ)	0.4
$r_{2,2}$	OR	IF (x *is* PS) THEN (o *is* PS)	0.6
$r_{3,2}$	OR	IF (x *is* PM) THEN (o *is* PS)	0.2
$r_{3,3}$	OR	IF (x *is* PM) THEN (o *is* PM)	0.8
$r_{4,4}$	OR	IF (x *is* PL) THEN (o *is* PL)	1.0

This produces a fuzzy relational surface which is piecewise linear between rule centres and the general trend of the input, output relationship, almost a linear mapping, is obvious from the contour plot.

When the input is known, the fuzzy inferencing algorithms produce a single fuzzy output set for each rule, and the output of the fuzzy rule base is the union of all these membership functions defined on the output universe. It should therefore be clear that when a fuzzy algorithm is implemented, fuzzy membership function intersection generally occurs across different universes whereas fuzzy union usually takes place on the same universe.

3.7 Inferencing

Inferencing is the process of reasoning about a particular state, using all available knowledge to produce a best estimate of the output. In a fuzzy system, the inference engine is used to pattern match the current fuzzy input set $\mu_A(\mathbf{x})$ with the antecedents of all the fuzzy rules and to combine their responses, producing a single fuzzy output set $\mu_B(o)$. This is defined by:

$$\mu_B(y) = \widehat{\sum}_{\mathbf{x}} \left(\mu_A(\mathbf{x}) \hat{*} \mu_R(\mathbf{x}, o)\right) \tag{51}$$

where the triangular co-norm $\widehat{\sum}_{\mathbf{x}}$ is taken over *all* possible values of $\mathbf{x}$, and the triangular-norm computes a match between two membership functions for a particular value of $\mathbf{x}$. When $\widehat{\sum}$ and $\widehat{\prod}$ are chosen to be the integration (sum) and the product operators, respectively, then:

$$\mu_B(y) = \int_D \mu_A(\mathbf{x})\, \mu_R(\mathbf{x}, o)\, d\mathbf{x} \tag{52}$$

which for an arbitrary fuzzy input set requires an n-dimensional integral to be evaluated over the input domain D. The calculated fuzzy output set depends on the fuzzy input set $\mu_A(.)$, the relational surface $\mu_R(.)$ as well as the actual inferencing operators.

As long as there exists an overlap between the fuzzy input set and the antecedents of the rule base, then the fuzzy system is able to *generalise* in some sense. The ability to generalise information about neighbouring states is one of the strengths of fuzzy logic, but their actual interpolation properties are poorly understood. The neurofuzzy systems studied in this report are particularly important as their approximation abilities can be both determined and analysed theoretically which has many important consequences for practical systems.

3.8 Fuzzification and Defuzzification

The fuzzy membership functions are the interface between the real-valued world outside the fuzzy system and its own internal rule-based representation. Hence, a real-valued input must be represented as a fuzzy set in order to perform the inferencing calculations and the information contained in the fuzzy output set must be compressed to a single number which is the real-valued output of the fuzzy system. This section discusses different methods for performing these operations.

3.8.1 Fuzzification

The process of representing a real-valued signal as a fuzzy set is known as *fuzzification* and is necessary when a fuzzy system deals with real-valued inputs. There are many different methods for implementing a fuzzifier but the most commonly used is the *singleton* that maps the input $\mathbf{x}$ to a *crisp* fuzzy set with membership:

$$\mu_{\tilde{\mathbf{x}}}(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} = \tilde{\mathbf{x}} \\ 0 & \text{otherwise} \end{cases} \tag{53}$$

For inputs that are corrupted by noise, the shape of the fuzzy set can reflect the uncertainty associated with the measurement process. For example, a triangular fuzzy set may be used where the vertex corresponds to the mean of some measurement data and the base width is a function of the standard deviation. If the model input is a linguistic statement, a fuzzy set must be found that adequately represents this statement. Unless the input is a linguistic statement, there is *no* justification for fuzzifying the input using the same membership functions used to represent the linguistic statements such as *x is small*. The latter membership functions are chosen to represent *vague* linguistic statements whereas the input fuzzy sets reflect the uncertainty associated with the *imprecise* measurement process, and these two quantities are generally distinct. A fuzzy input distribution effectively low pass filters or averages neighbouring outputs and as the width of the input set grows (increasingly imprecise measurements), a greater emphasis is placed on neighbouring output values and the system becomes more conservative in its recommendations [5].

3.8.2 Defuzzification

When a fuzzy output set $\mu_B(o)$ is formed as the output of the inferencing process, it is necessary to compress this distribution to produce a single value, representing the output of the fuzzy system. This process is known as defuzzification and currently there are several commonly used methods. Perhaps the two most widely used are the Mean of Maxima (MOM) and the Centre of Gravity (COG) algorithms which are illustrated in figure 28. These can be classed as truncation and algebraic defuzzification methods, respectively, as the former bases the output estimate on only one piece of information (or at most an average of several) because the output is the value which has the largest membership in $\mu_B(o)$, whereas the latter uses the normalised weighted contribution from every point in the output distribution. The COG defuzzification algorithm tends to give a smoother output surface as there is a more gradual transition between the rules as the input is varied.

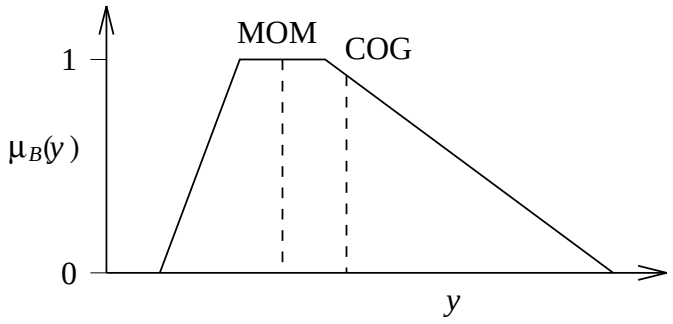


Figure 28: The Mean of Maxima and Centre of Gravity defuzzification algorithms.

The COG defuzzification process is defined by:

$$o(\mathbf{x}) = \frac{\int_O \mu_B(o)\, o\, do}{\int_O \mu_B(o)\, do} \tag{54}$$

and the whole of the output distribution contributes to determining the network's output. This is in direct contrast with the MOM procedure where only the elements with maximal membership are considered and the rest of the distribution is taken as being unimportant. This can be expressed as:

$$o(\mathbf{x}) = \frac{\int_O \mu_{B^H}(o)\, o\, do}{\int_Y \mu_{B^H}(o)\, do} \tag{55}$$

where $\mu_{B^H}(o)$ is the fuzzy set obtained by taking the α-cut at the height, H_B, of B.

Just as there exists a whole family of T-norms and S-norms, there is a large number of defuzzification algorithms. In practice though, the COG defuzzification procedure is most widely used, for reasons that will be explained in the next section.

4 Fuzzy Systems

A *fuzzy system* contains all the components necessary to implement a fuzzy algorithm and resolve all of the associated vagueness. It is composed of four basic elements:

- **a knowledge base** which contains definitions of the fuzzy sets and the fuzzy operators;
- **an inference engine** which performs all the output calculations;
- **a fuzzifier** which represents the real valued inputs as fuzzy sets; and
- **a defuzzifier** which transforms the fuzzy output set to a real valued output,

and this is illustrated in figure 29. The knowledge base contains the definitions of each of the fuzzy sets and maintains a store of operators used to implement the underlying logic (AND, OR etc.), as well as a *rule confidence matrix* which represents the fuzzy rule mappings. The inference unit, together with the fuzzifier and the defuzzifier allows real-valued outputs to be calculated from real valued inputs. The fuzzifier represents the input as a fuzzy set which allows the inferencing unit to match it against the antecedents of the rules stored in the knowledge base. Then the inferencing unit calculates how strongly each rule fires and outputs a fuzzy distribution (union of all the fuzzy output sets) that represents its fuzzy estimate of the true output. Finally, this information is defuzzified (compressed) into a single value which is the output of the fuzzy system.

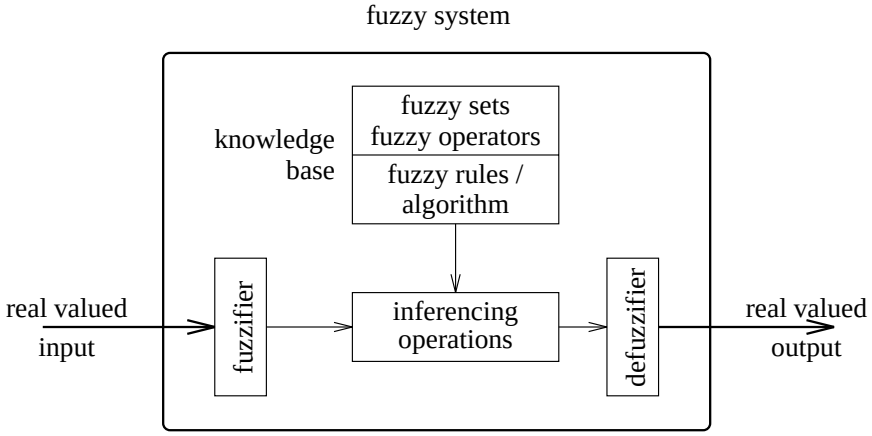


Figure 29: A fuzzy system is composed of a knowledge base, an inference engine, a fuzzifier and a defuzzifier.

These systems are extremely flexible and can be used as a basic plant model, a controller, an estimator, or to represent a performance function or as a desired trajectory generator. They implement a general nonlinear mapping and as such can be used for many approximation or classification tasks depending on how the inputs and outputs are chosen. Fuzzy systems have a fuzzy algorithm as their knowledge base, but once it's implemented on a computer, any vagueness is resolved and the mapping is completely deterministic and in some cases has quite a simple mathematical representation.

4.1 Functional Mapping

Many engineering applications require a fuzzy system that simply operates as a functional mapping, mapping real-valued inputs to real-valued outputs where the task is to approximate a function $o = f(\mathbf{x})$ on a bounded area (*compact*) of the input space. In contrast to the data-driven methods used to train ANNs, fuzzy systems are designed using human-centred engineering techniques where the system is used to encode the heuristic knowledge articulated by a domain-specific expert. A finite number of vague or fuzzy rules forms the basis for the fuzzy system's knowledge base and to *generalise* or interpolate between these rules, the inference engine weights each rule according to its firing strength, which in turn is determined by both the shape of the fuzzy membership functions and the logical operators used by the inference engine. This section shows that when a centre of gravity defuzzification algorithm is used in conjunction with algebraic operators, then the type of functional mapping performed by the system is directly dependent on the shape of the fuzzy input sets. The rule confidence matrix is a set of parameters that determines the magnitude (height) of the fuzzy mapping, but it is the fuzzy input sets that determine its form.

4.1.1 Analysis

Consider a fuzzy system that uses a centre of gravity defuzzification algorithm, then the network's output is: given by:

$$o(\mathbf{x}) = \frac{\int_O \mu_B(o)\, o\, do}{\int_O \mu_B(o)\, do}. \tag{56}$$

When the T-norm and S-norm operators are implemented using product and sum functions, respectively, then the centre of gravity defuzzification algorithm becomes:

$$o(\mathbf{x}) = \frac{\int_O \int_{\mathrm{X}} \mu_A(\mathbf{x}) \sum_{ij} \mu_{A^i}(\mathbf{x})\, \mu_{B^j}(o)\, c_{ij}\, o\, d\mathbf{x}\, do}{\int_O \int_{\mathrm{X}} \mu_A(\mathbf{x}) \sum_{ij} \mu_{A^i}(\mathbf{x})\, \mu_{B^j}(o)\, c_{ij}\, d\mathbf{x}\, do}. \tag{57}$$

But for bounded and symmetric fuzzy output sets the integrals $\int_O \mu_{B^j}(o)\, do$, for all j, are equal and so the following relationship holds:

$$\frac{\int_O \mu_{B^j}(o)\, o\, do}{\int_O \mu_{B^j}(o)\, do} = o_j^c$$

where o_j^c is the centre of the j^{th} output set, and equation 56 therefore reduces to:

$$o(\mathbf{x}) = \frac{\int_{\mathrm{X}} \mu_A(\mathbf{x}) \sum_i \mu_{A^i}(\mathbf{x}) \sum_j c_{ij}\, o_j^c\, d\mathbf{x}}{\int_{\mathrm{X}} \mu_A(\mathbf{x}) \sum_i \mu_{A^i}(\mathbf{x}) \sum_j c_{ij}\, d\mathbf{x}}.$$

Suppose that the multivariate fuzzy input sets form a partition of unity, ie. $\sum_i \mu_{A^i}(x) \equiv 1$ and that the i^{th} rule confidence vector $\mathbf{c}_i = (c_{i1}, \ldots, c_{iq})^T$ is normalised, ie. $\sum_j c_{ij} \equiv 1$, then the defuzzified output becomes:

$$o(\mathbf{x}) = \frac{\int_{\mathrm{X}} \mu_A(\mathbf{x}) \sum_i \mu_{A^i}(\mathbf{x}) w_i\, d\mathbf{x}}{\int_{\mathrm{X}} \mu_A(\mathbf{x})\, d\mathbf{x}} \tag{58}$$

where $w_i = \sum_j c_{ij}\, o_j^c$ is the *weight* associated with the i^{th} fuzzy membership function. The transformation from the weight w_i to the vector of rule confidences $\mathbf{c}_i$ is a one-to-many mapping, although for fuzzy sets defined by symmetric B-splines of order $r \geq 2$, it can be inverted in the sense that for a given w_i there exists a *unique* $\mathbf{c}_i$ that will generate the desired output. This will be explained further in section 4.1.2. It should also be emphasised that using weights in place of rule confidence vectors provides a considerable reduction in both

the storage requirements and the computational cost, and is also relevant to the discussion on training given in section 5.2.

When the fuzzy input set $\mu_A(\mathbf{x})$ is a singleton, the numerator and denominator integrals in equation 58 cancel to give

$$o^s(\mathbf{x}) = \sum_i \mu_{A^i}(\mathbf{x})\, w_i \tag{59}$$

where $o^s(\mathbf{x})$ is called the fuzzy singleton output. This is an important observation since $o^s(\mathbf{x})$ is a *linear* combination of the fuzzy input sets and does *not* depend on the choice of fuzzy output sets. It also provides a useful link between fuzzy and neural networks and allows both approaches to be treated within a unified framework, and this is discussed in section 5. The reduction in the computational cost of implementing a fuzzy system in this manner and the overall algorithmic simplification is illustrated in figure 30.

The analysis also illustrates how the centre of gravity defuzzification procedure *implicitly* imposes a partition of unity on the fuzzy input membership functions. Consider the above system when the fuzzy input sets do not sum to unity, which could be due to their univariate shape or the operator used to represent fuzzy intersection. The output is then given by:

$$\begin{aligned} o^s(\mathbf{x}) &= \frac{\sum_i \mu_{A^i}(\mathbf{x})\, w_i}{\sum_j \mu_{A^j}(\mathbf{x})} \\ &= \sum_i \mu_{\hat{A}^i}(\mathbf{x})\, w_i \end{aligned} \tag{60}$$

where the normalised fuzzy input membership functions $\mu_{\hat{A}^i}(\mathbf{x}) \left(\mu_{A^i}(\mathrm{x}) / \sum_j \mu_{A^j}(\mathrm{x})\right)$ form a partition of unity. This normalisation step is very important because it determines the *actual* influence of the fuzzy set on the system's output and can make previously convex sets, non-convex.

When the input to the fuzzy system is a fuzzy distribution rather than a singleton, it is possible to substitute equation 59 into 58 giving:

$$o(\mathbf{x}) = \frac{\int_{\mathrm{X}} \mu_A(\mathbf{x})\, o^s(\mathbf{x})\, d\mathbf{x}}{\int_{\mathrm{X}} \mu_A(\mathbf{x})\, d\mathbf{x}}. \tag{61}$$

The defuzzified output is a weighted average of the fuzzy singleton outputs over the support of the fuzzy input set $\mu_A(\mathbf{x})$, and the effect is to *smooth* or *low pass filter* the system's output, o. This is illustrated in figure 31. It can be seen that as the width of the fuzzy input set increases, the overall output of the system becomes *less sensitive* to the shape of either the input set or the sets used to represent the linguistic terms. However this is not always desirable as the output also becomes less sensitive to individual rules and the input variable, and in the limit as the input set shape has an arbitrarily large width (representing complete uncertainty about the measurement) the system's output will be constant everywhere.

An important consequence of the above analysis is that using centre of gravity defuzzification in conjunction with the sumand productoperators has reduced fuzzy composition and defuzzification to a single operation. It is no longer necessary to calculate and store $\mu_R(\mathbf{x}, o)$.

4.1.2 Rule Confidences and Weights

The simple relationship between a single weight w_i, the corresponding rule confidence vector $\mathbf{c}_i$ and the fuzzy output membership function illustrates their role when a fuzzy algorithm is implemented as a fuzzy system. The weight w_i can be interpreted as being a local estimate of the output of the system given that the input lies in the corresponding fuzzy input membership function $\mathbf{A}^i$, and this can be expressed linguistically as:

$$\text{IF } (\mathbf{x} \textit{ is } \mathbf{A}^i) \text{ THEN } (o \textit{ is } w_i) \tag{62}$$

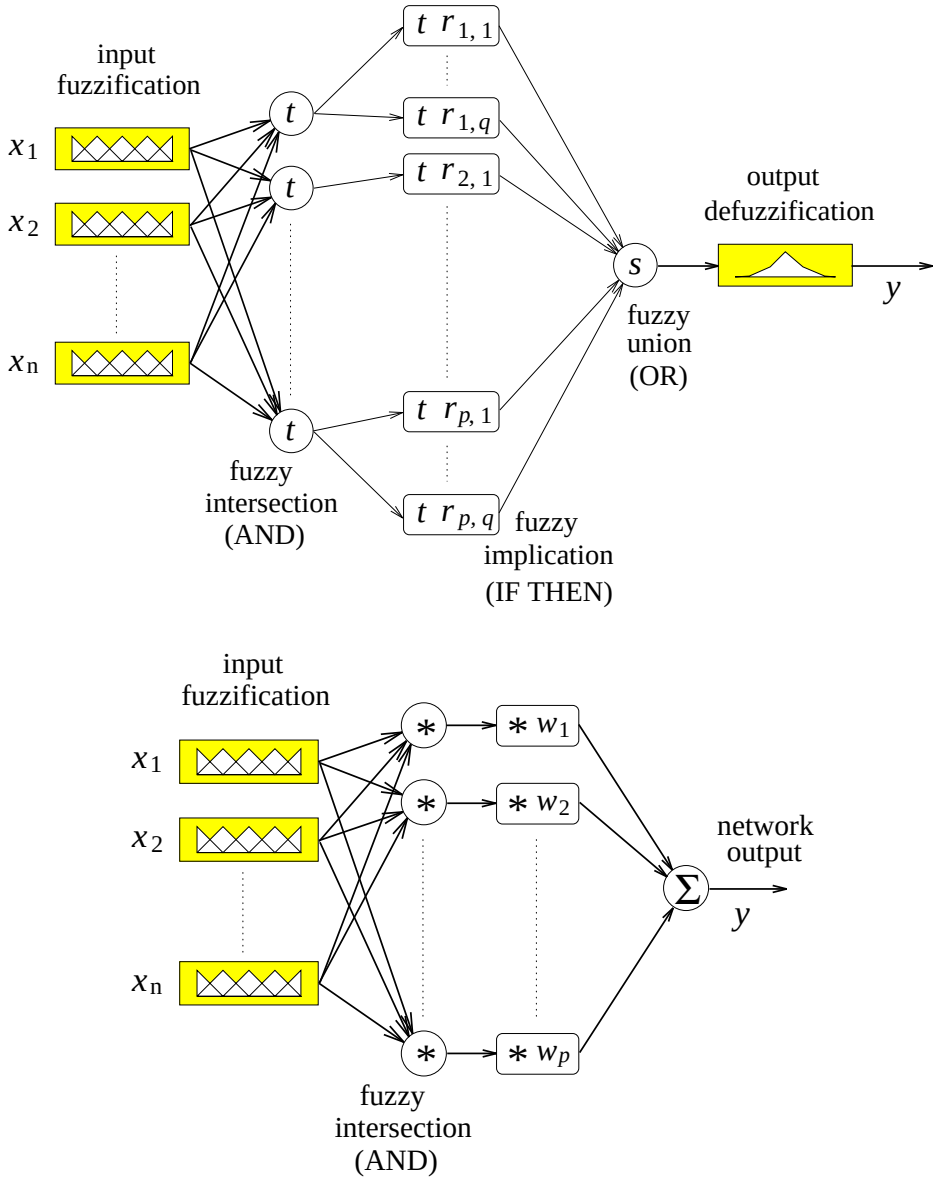


Figure 30: An illustration of the information flow through a fuzzy system (top) and the resulting simplification (bottom) when algebraic operators are used in conjunction with a centre of gravity defuzzification algorithm, and the singleton input is represented by a crisp fuzzy set.

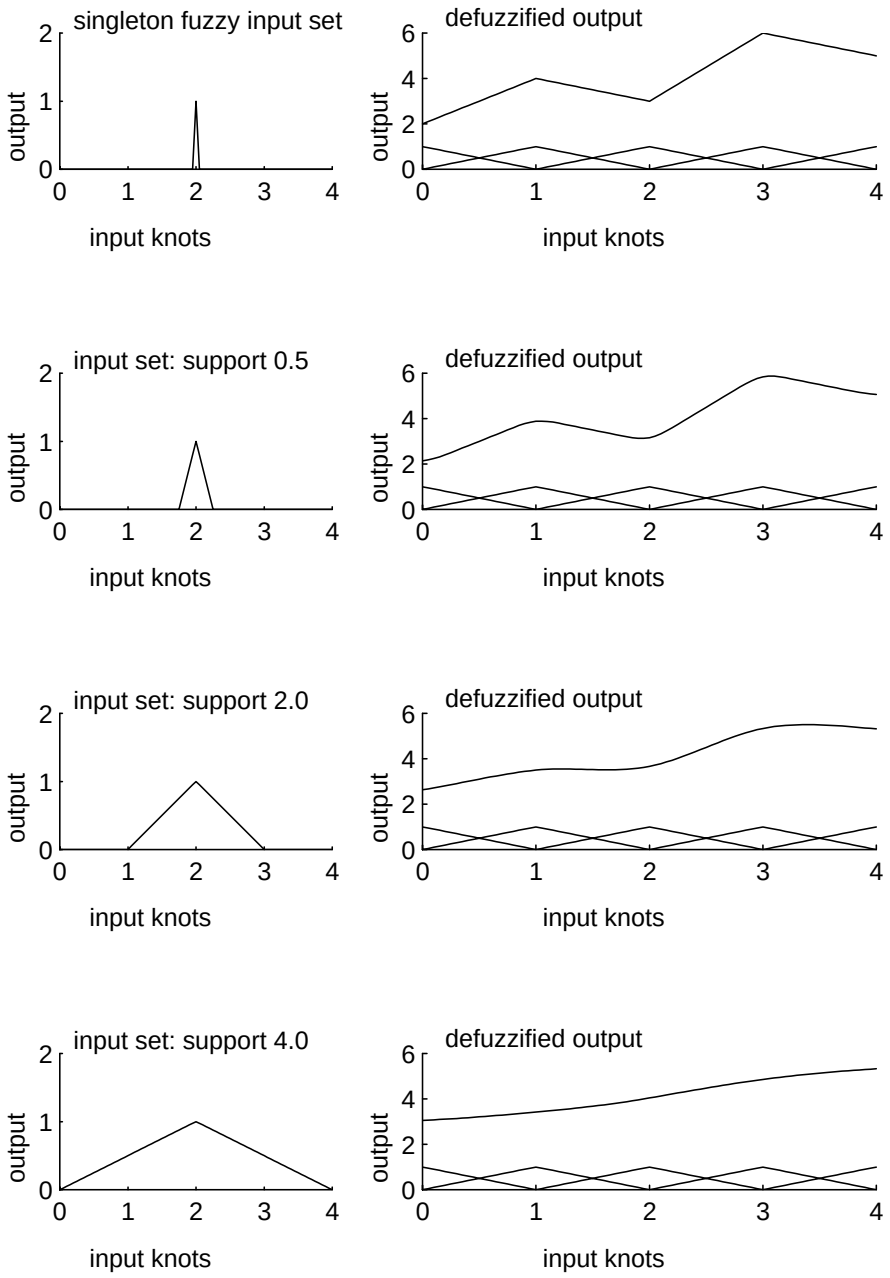


Figure 31: Four fuzzy input sets and their corresponding defuzzified outputs, when the fuzzy rule base consists of triangular membership functions. The original triangular membership functions used to represent the linguistic terms are shown on the bottom of the graphs on the right, and it can clearly be seen that as the width of the input set increases the system becomes less sensitive to the input variable and the set shapes.

Therefore the fuzzy rule confidences and the fuzzy output membership functions simply provide a linguistic-based technique for setting these weights. Despite the fact that a fuzzy algorithm is composed of vague, linguistic terms, the defuzzification algorithm reduces each rule confidence vector to a single numerical value: the weight. For example, consider part of the fuzzy algorithm shown in figure 27:

$$\begin{array}{ll} \text{IF } (x \textit{ is } PS) \text{ THEN } (o \textit{ is } AZ) & 0.4 \\ \text{IF } (x \textit{ is } PS) \text{ THEN } (o \textit{ is } PS) & 0.6 \end{array}$$

where the output sets AZ and PS are centred on 0 and 1, respectively. Specifying these two rules is equivalent to saying that:

$$\text{IF } (\mathbf{x} \textit{ is } \mathbf{A}^i) \text{ THEN } (o \textit{ is } 0.6) \tag{63}$$

except that for the former, it is arguably more natural for an expert to express both inputs and outputs as linguistic terms. However, it should be emphasised that the expert is really providing a precise value, even though he has expressed a set of actions using vague terminology. Sometimes, researchers confuse the issues and talk about a fuzzy singleton output set centred on the numerical value. This is numerically consistent as the output of a network configured like this would be the same as one that stored numerical values, although in reality, linguistic terminology is only useful for initialising and validating the neurofuzzy system. Saying that the output is a singleton fuzzy set centred on a value of 0.6 is no more helpful than saying that the corresponding numerical value *is* 0.6. Linguistic inputs and output sets are only useful because they speak the same language as an expert.

The mapping from rule confidences to weights is also invertible in the sense that given a system which composed of rules such as that in 62, a fuzzy algorithm with rule confidences and linguistic outputs can be generated. This is possible because w_i can be interpreted as a local estimate of the network's output, therefore it is consistent to evaluate its degree of membership in the fuzzy output sets, and assign this to the corresponding rule confidence:

$$c_{ij} = \mu_{Bj}(w_i) \tag{64}$$

In [5], it is shown that no information is lost when this transformation is made (using symmetric B-splines as the fuzzy output membership functions) as when the corresponding rule confidence vector is defuzzified, the original weight is obtained. Hence, a fuzzy system which uses a centre of gravity defuzzification algorithm and algebraic operators can be implemented using the reduced form shown in equation 59 and still have a linguistic interface for initialisation and validation purposes because of the invertible mapping that exists between weights and rule confidences.

4.2 Factors affecting the Functional Mapping

Having derived the very simple relationship between the fuzzy input sets and the network's output, as well as the one between the weights and rule confidences, the following section investigates some of implications of this observation and also looks at slightly differing implementation strategies.

As an illustration, consider the following fuzzy algorithm which is composed of 3 rules:

$$\begin{array}{ll} & \text{IF } (x \textit{ is small}) \text{ THEN } (o \textit{ is small}) \\ \text{OR} & \text{IF } (x \textit{ is medium}) \text{ THEN } (o \textit{ is medium}) \\ \text{OR} & \text{IF } (x \textit{ is large}) \text{ THEN } (o \textit{ is small}) \end{array}$$

forms part of a knowledge base in two systems using:

1. triangular fuzzy sets and algebraic operators, and

2. Gaussian fuzzy membership functions and truncation operators.

The rule confidence matrix is *binary* and hence each fuzzy rule either totally fires or else is completely inactive. Similarly, the fuzzy algorithm has only one input and so the only fuzzy operations on the fuzzy membership functions are *implication* and *union*.

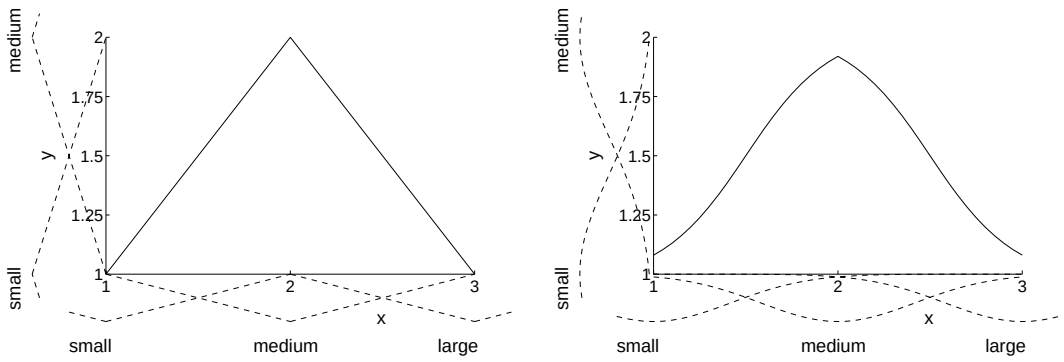


Figure 32: A comparison between a fuzzy system based on triangular membership functions and algebraic operators (left) and one that uses Gaussian membership functions and truncation operators (right). The appropriate membership functions are shown along the respective axes.

From the two diagrams shown in figure 32, it should be clear that the outputs of the two systems are similar, but different. The triangular/algebraic fuzzy system simply joins the rule centres with straight lines which is because the triangular membership functions are simply straight lines on each interval. The Gaussian fuzzy system has an output which is "more curved" between the set centres and has a sharp peak in the centre. It is interesting to note that this peak is due to the choice of truncation operators, as a similar system that uses algebraic operators is shown in figure 33.

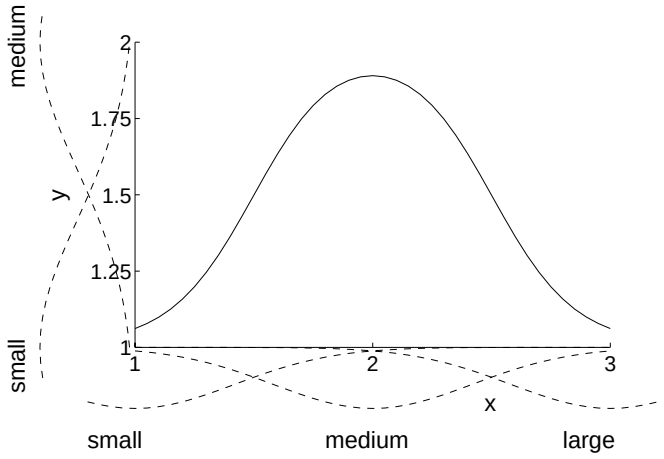


Figure 33: The output of a Gaussian fuzzy system with algebraic operators.

Therefore, it should be clear that the type of decision surface that is formed by the fuzzy system depends on the fuzzy *algorithm*, the fuzzy *variables* and the fuzzy *operators*, and these points will be further discussed in the following sections.

4.3 Algebraic Operators

In section 3, two families of operators were introduced (T-norms and S-norms) and it was shown that there exist many simple functions which can belong to each class. It is impossible to say that one operator will always be better than another as this very much depends on the available knowledge and the form of the underlying mapping.

The effect of different fuzzy operators will be described first, as they play a major role in determining the form of the fuzzy system's output.

4.4 Fuzzy Membership Functions

It has been shown that when algebraic fuzzy operators are used, the form of the fuzzy decision surface is directly related to the shape of the fuzzy input membership functions, and this is illustrated in figures 15 and 32 where the output of the fuzzy system is formed from a linear combination of the membership functions. This comparatively simple but pertinent observation reopens the debate about how fuzzy membership functions should be chosen and again illustrates how there may be a potential conflict between modelling and representational requirements.

For algebraic fuzzy operators, the form of the fuzzy surface is directly related to the shape of the fuzzy input membership functions, and if the normalised fuzzy variable (one which forms a partition of unity) is piecewise constant or linear on an interval in the input space, the network's output will also be piecewise constant and linear, respectively. This is illustrated using trapezoidal fuzzy membership functions in figure 15. The shape of the fuzzy membership function locally determines the form of the decision surface and this should be taken into account when they are designed, as discussed in section 2.6.2.

4.4.1 Locally Constant Membership Functions

The direct relationship between the shape of the fuzzy input membership functions and the decision surface may initially seem intuitive but consider what implications it has for *Gaussian* and Π fuzzy membership functions. A Π fuzzy membership function is described by:

$$\mu_A(x) = \begin{cases} 0 & \text{for } |x-c| \geq 2\sigma \\ \frac{(c-x)^2 - 4\sigma^2}{6\sigma^2} & \text{for } \sigma \leq |x-c| < 2\sigma \\ 1 - \left(\frac{x-c}{c-\sigma}\right)^2 & \text{otherwise} \end{cases} \tag{65}$$

where c represents the centre of the membership function and σ is its width (distance from the centre to a membership value of 0.5), and its form is illustrated in figure 8. This fuzzy membership function, like a Gaussian, has a zero derivative at its centre, therefore unless the sets overlap significantly at the centre and another membership function has a non-zero derivative at this point, it is difficult to model linear functions in this region as the decision surface will be almost constant. Π fuzzy membership functions are generally arranged like triangular sets, with at most two overlapping at any one time and at the centre of a rule, only one is non-zero. This causes problems as at a rule's centre, every membership function would have a zero derivative and the fuzzy decision surface would always be locally constant around this point, and its form would resemble a series of plateaus. How noticeable this is for Gaussian fuzzy membership functions depends on the degree of overlap as they do not have a strictly compact support, but this effect can be seen when the widths are chosen inappropriately, as illustrated in figure 34. In this figure, the output of a Gaussian fuzzy system is plotted together with its membership functions and their normalised counterparts which form a partition of unity. The overall form of the mapping is a $\vee$-shape with a

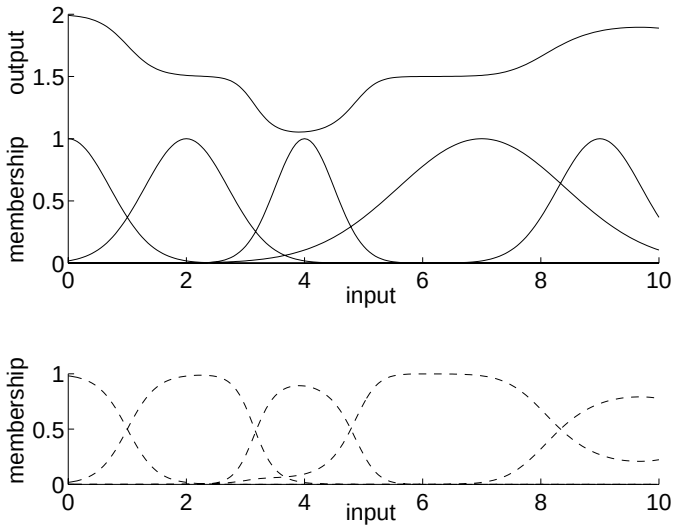


Figure 34: The output of a fuzzy system when the Gaussian fuzzy membership functions' widths are badly chosen. Also shown are the normalised fuzzy membership functions on the bottom figure.

downwards trend until the centre of the third basis function and an upwards slope thereafter. The locally constant regions can easily be identified, roughly corresponding to the centres of the second and fourth membership function, and this could have been predicted by looking at the shape of the normalised membership functions as they have a similar structure.

For many engineering applications, a set of locally constant regions around each rule centre is undesirable, as the system's output will change rapidly in some regions and not about a rule's centre. For most systems, it would be illogical to require the output to be locally constant around a rule centre and such as network would not even be able to approximate *linear* functions satisfactorily. Hence, the fuzzy membership functions should be chosen such that these piecewise constant regions are either explicit in its shape (using trapezoidal functions) or else they overlap sufficiently so that there is the potential to model locally linear behaviour about a rule's centre.

4.4.2 Normalised Fuzzy Variables

One of the main points from the last section which discussed locally constant membership functions must be that the normalised fuzzy variables which form a partition of unity are important in predicting the form of the functional mapping, especially when the original fuzzy membership functions do not possess this property. From equation 59, it can be seen that the network's output is always formed from a linear combination of the *normalised* fuzzy membership functions, hence even when the original fuzzy membership functions do not form a partition of unity, the network will implicitly perform a normalising operation before calculating the output. The importance of understanding this property is illustrated in figures 8 and 11 where normalised and unnormalised membership functions are displayed. The normalised fuzzy variables have piecewise constant regions whenever only one rule contributes to the output and the piecewise linear regions also mean that the network's output will also be piecewise linear in this region. These, and other, factors may not be immediately obvious from the original unnormalised fuzzy membership functions.

Choosing the shape of the fuzzy membership functions *biases* the network as it effectively determines which mappings the network can and cannot effectively approximate. When the

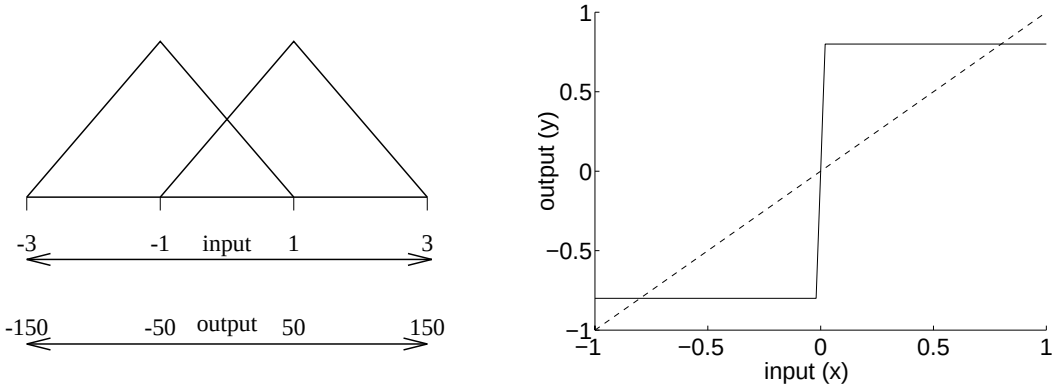


Figure 35: The almost discontinuous fuzzy decision surface caused by the interaction between truncation operators and rule confidences. The equivalent algebraic system's output is shown as a dashed line.

membership functions do not form a partition of unity, then the designer has little understanding about how they have biased the network's structure. Calculating the normalised fuzzy variables will not change the network's mapping ability, but it does provide extra insight into the influence of the membership function and rule on the overall system's output.

4.5 Fuzzy Algorithms and Rule Confidences

The fuzzy algorithm obviously has an influence on the actual output of the fuzzy system, but sometimes the rule confidences can interact with truncation operators to produce strange behaviours. This can be illustrated with the following example.

Consider a single input, single output fuzzy system which has the following four rules in its knowledge base:

	IF (x *is small*) THEN (o *is negative large*)	0.51
OR	IF (x *is small*) THEN (o *is positive large*)	0.49
OR	IF (x *is large*) THEN (o *is negative large*)	0.49
OR	IF (x *is large*) THEN (o *is positive large*)	0.51

where the fuzzy membership functions and the output surface is shown in figure 35.

Despite the fact that piecewise linear input membership functions are defined on the interval $[-1, 1]$, the output surface is nearly discontinuous at the point 0. This is caused by the interplay between the rule confidences and the truncation implication operator and its effect is magnified because the output sets are large distances apart (but still overlapping). The use of the max operator to represent fuzzy union (ORing) also causes the output to be -0.8 and 0.8 at the centres of the two input sets, whereas using the addition operator produces an output of -1 and 1, respectively. When the algebraic product operator is used to represent implication as well, the output is simply a straight line between -1 and 1 as would be expected and this is also illustrated in figure 35.

4.6 Discussion

With the countless number of ways for implementing a fuzzy system, it is difficult to prove or say anything meaningful about their generalisation properties and to use this information to develop new construction and learning algorithms. It is trivial to show that fuzzy systems

are *universal approximators*[4], although so are look-up tables and the power of a fuzzy system is contained in its ability to represent human expertise and to locally generalise (interpolate and extrapolate) the information smoothly. It is impossible to say that one type of fuzzy system will always perform better than another type, as it would always be possible to find (construct) a dataset that was of the same form as the output of any fuzzy system. However, researchers and practitioners should focus their attention on those implementations that are *transparent*.

A fuzzy system is transparent when its internal workings are clearly understood by the designer. This means that when a fuzzy algorithm is stored in the knowledge base, the designer has a good understanding of how the fuzzy system will generalise between rules. It has been shown that the truncation operators can produce some strange, nearly unpredictable sharp nonlinearities due to subtle interplays between individual rules. Using algebraic operators results in a fuzzy system whose form is directly related to the shape of the input membership functions. Therefore its interpolation abilities can be as simple as:

joining the dots with straight lines

when triangular fuzzy membership functions are used, and smoother curves when quadratic (and higher-order) B-splines and Gaussian membership functions are employed.

These algebraic fuzzy networks have the structure of a 3 or 5-layer artificial neural network hence they are termed *neurofuzzy systems*. These systems and their implementation will be discussed in the remainder of this report.

5 Neurofuzzy Networks

Neurofuzzy systems are currently one of the "flavours of the month" in the neural network and fuzzy logic communities. They attempt to combine the structural and learning abilities of a neural network with the linguistic initialisation and validation aspects of a fuzzy system. Neurofuzzy networks are a particular type of fuzzy system that uses algebraic operators and continuous fuzzy membership functions, as it has been shown that these are generally the best for surface fitting problems. There are several types of neurofuzzy system, and these are categorised by the type of membership functions; the two most common are *B-splines* and *Gaussians*. By regarding these fuzzy systems as types of neural networks, the role of the membership function in determining the *form* of the decision surface is highlighted, rather than its accuracy in modelling the vagueness or uncertainty associated with a particular linguistic term. This interpretation also provides several possibilities for utilising inductive learning-type techniques to produce parsimonious rule bases. Thus the mathematical rigour associated with adaptive neural networks can be used to analyse the behaviour of learning neurofuzzy systems and improve their performance as data-driven and human-centred approaches are being combined to improve the network's overall performance.

Originally, ANNs were "sold" as black-box learning systems and they've had a considerable amount of success in modelling and controlling ill-defined problems. However, users are beginning to demand a greater degree of understanding of the network's performance and structure as these networks are beginning to be applied in safety critical systems and also designers always want to improve the performance. It is only possible to improve the performance of a system manually, if the designer has a good insight into how changing a particular parameter affects the overall output. The rule-based representation of neurofuzzy systems offers this *transparency*.

[4] A universal approximator can model any continuous nonlinear function to any desired degree of accuracy on a compact domain given sufficient resources.

5.1 Architecture

Neurofuzzy systems can be regarded as particular types of fuzzy systems where the input is represented as a singleton, and centre of gravity and algebraic fuzzy operators are used in the inferencing calculations. Therefore the output of such a network is calculated using equation 59 and is very simple to implement in software or hardware.

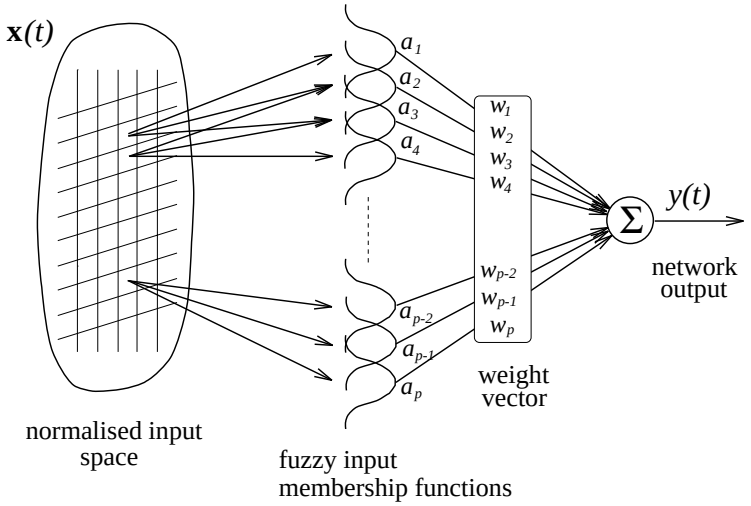


Figure 36: A typical neurofuzzy network where the fuzzy input membership functions are distributed across the input space in a regular fashion and the output is formed from a linear combination of these functions.

Neurofuzzy systems have a particular structure and can be drawn as either 3 or 5-layer networks depending on the level of detail. In figure 36, the network's functionality is separated into 2 distinct operations. The first is the input to multivariate fuzzy input set mapping represented as:

$$f : \mathbf{x} \rightarrow \mathbf{a} \tag{66}$$

where $\mathbf{a} = (\mu_{\mathrm{A}^1}(\mathbf{x}), \ldots, \mu_{\mathrm{A}^p}(\mathbf{x}))$ is the p-dimensional vector containing the outputs of all the multivariate fuzzy input sets. The second transformation is simply a (normalised) linear sum:

$$g : \mathbf{a} \rightarrow o \tag{67}$$

where:

$$o = \frac{\sum_{i=1}^{p} a_i w_i}{\sum_{j=1}^{p} a_j} \tag{68}$$

The second transformation is identical for each of the neurofuzzy systems as it is only the type of fuzzy input membership function which distinguishes the different architectures. This will be emphasised when the B-spline and the Gaussian neurofuzzy systems are described. All the neurofuzzy systems can therefore be represented using linguistic rules such as 62, and can be given a full linguistic interpretation because of the invertible relationship that exists between the weights and rule confidences.

This decomposition of the output calculation into two separate mappings also illustrates how these networks can be trained. As the second mapping g is simply a linear combination of the fuzzy input membership functions, any of the adaptive linear training algorithms that has been developed over the past 30 years can be used to train the weights. It is also worthwhile re-emphasising that a rule confidence matrix can be generated from the weight vector, hence any of these adaptive neurofuzzy networks can be represented as a fuzzy linguistic algorithm.

The first mapping f is highly nonlinear as it is described by the type and position of the fuzzy input membership functions. However, the dual interpretation of the network as either an ANN or a fuzzy system means that it is possible to use either inductive learning-type algorithms to train these basis functions or else used supervised or unsupervised "neural" training rules.

5.1.1 B-splines

When the membership functions in a fuzzy variable are represented using univariate B-splines of order k (see section 2.3.1), a centre of gravity defuzzification algorithm is used, algebraic fuzzy reasoning operators are employed together with a singleton fuzzification procedure, the output of the fuzzy system is given in equation 59. This is equivalent to a standard B-spline (neural) network that is often used for surface fitting applications where the fuzzy input sets are known as *basis functions*, and the set of linear weights are calculated using standard matrix pseudo-inversion routines. Using the terminology basis functions instead of (multivariate) fuzzy input sets also emphasises that they determine the form of the network's decision surface, and these two terms will both be used for the remainder of this report.

This link between a conventional surface fitting algorithm and the class of fuzzy systems is important both for implementing fuzzy systems and for analysing their performance theoretically. It is trivial to show that these systems are *universal approximators*, as a B-spline neurofuzzy system produces piecewise polynomial mappings and by the Stone-Weierstrass theorem, the set of polynomial functions are universal approximators. It is also worth noting that by increasing either the number of the B-spline fuzzy sets *or* the order of the splines, it is possible to prove this result. This interpretation is also important because, as is described in section 5.2, the set of adjustable parameters (weights) is *linear* and there exists a lot of standard learning theory that can be applied to analyse the behaviour of these adaptive systems [5]. It also means that there is a considerable reduction in the computational cost of the system when it is implemented as described in equation 59 rather than performing the full fuzzification $\rightarrow$ inference $\rightarrow$ defuzzification calculation. Finally, and perhaps most importantly, this interpretation allows more advanced neurofuzzy learning algorithms to be developed which can exploit redundancy in the training data and be applied to high-dimensional modelling and control problems.

As was described in section 3.4.2, the multivariate B-splines are formed from the *tensor product* of the univariate ones (this means that every possible combination of univariate basis functions is multiplied together), hence they are evenly distributed across a lattice. This can cause problems that are well-known in the surface fitting community and the main ones are:

Curse of Dimensionality where the number of basis functions required to populate the input space grows exponentially with respect to the input space dimension.

Irregular Data where there are holes in the training data which make it impossible to directly determine some of the weights, as there is no data in the support of the basis function.

Ridge Functions as it is difficult to approximate ridge functions which do not run parallel to an axis using lattice-based techniques.

These problems occur in all lattice-based fuzzy systems, but their effect is difficult to quantify because of the other algorithmic complexities such as truncation operators. Focusing on these problems directly though, means that potential solutions can be proposed such as using additive networks to overcome the curse of dimensionality (see section 6), using regularisation

techniques or non-lattice based rules to reduce the effect of irregular data and non-axis parallel knot lines can improve the quality of fit for ridge functions in certain cases [8].

In summary, B-spline neurofuzzy systems are extremely useful as they've been extensively developed as data-driven surface fitting algorithms and because the fuzzy linguistic interpretation means that experts can initialise and validate the network's behaviour. Also, the regular placement of the basis functions in the input space (on a lattice) allows simple addressing algorithms to be developed that allow real-time operation [5].

5.1.2 Gaussian Radial Basis Functions

When the fuzzy system is implemented as described previously, but with univariate membership functions represented as Gaussians, the fuzzy network is structurally identical to a *normalised* Gaussian Radial Basis Function (GRBF) algorithm, which is another technique that originated in the surface fitting community. GRBF networks provide an alternative to the lattice-based B-spline neurofuzzy systems, as they allow basis functions to be centred anywhere in the input space. The multivariate basis (fuzzy membership) functions are formed by multiplying each of the univariate basis functions together and this is equivalent to taking a Euclidean or elliptical measure (hence their name "radial") in the input space between the input and the basis function's centre and then passing this quantity through a univariate Gaussian as:

$$a_i(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{c}_i - \mathbf{x}\|_2^2}{2\sigma_i^2}\right) = \prod_{j=1}^{n} \exp\left(-\frac{(c_{ij} - x_j)^2}{2\sigma_i^2}\right) \tag{69}$$

This makes the link between radial basis functions and Gaussian fuzzy membership functions explicit [18, 34]. It also emphasises the fact that unimodal (multivariate) fuzzy membership functions can be regarded as an *inverse distance measure* (see section 2.2.2) where the distance between the set's centre and an input is calculated and this quantity is then passed through an appropriate (unimodal) nonlinearity. Unimodal fuzzy input sets simply partition up the input space and define what the network means by local.

The fuzzy systems considered in this report always use a COG defuzzification algorithm, so they implicitly impose a partition of unity on the multivariate Gaussian basis functions. This is an interesting feature that is not widely used in the GRBF community but it has been shown many times to improve the quality of the approximation [27], although sometimes the results can be unexpected as illustrated in figure 34.

Interpreting the multivariate fuzzy input membership functions as normalised Gaussian mappings has one important advantage in that it can be used to overcome the restriction that the basis functions' centres must be defined on an n-dimensional lattice. Various supervised and unsupervised learning rules can be used to select, optimise and cluster the fuzzy sets' centres, and therefore this representation is extremely useful when the data (training *and* testing) lie in a restricted part of the input space [34]. Gaussian functions are also *infinitely* differentiable and hence the fuzzy approximation and its derivatives of any order can be estimated (whether the model is trained accurately enough is another question though). Another important property is due to the local (but not strictly compact) support of the Gaussian basis functions. Normalising the output can make their supports appear global when only one basis function significantly contributes to the output, but almost compact when other basis function are defined near by. Hence the basic form is very flexible. A truly compact support, infinitely differentiable Gaussian-type basis function has also been proposed by Werntges [35], and this is shown in figure 10.

Despite these desirable numerical properties, the freedom in allowing the basis functions to be distributed anywhere in the input space often makes it difficult to interpret them using linguistic labels. The lattice-based B-spline basis functions can be interpreted as the

fuzzy intersection (AND) of the univariate fuzzy membership functions. However, there is no constraint on the placement of the multivariate basis functions, and projecting the univariate basis functions back onto each input axis could result in there being as many univariate input sets as multivariate input sets. It would be difficult to find that many linguistic terms to label each univariate set, so the usefulness of the fuzzy representation, in this case, is debatable.

5.2 Adaptive Systems

So far, this report has concentrated on static fuzzy and neurofuzzy systems where it has been assumed that both the network's structure and its internal rule base have been provided by the designer. The last part of this report will investigate how these features can be generated directly from training data, thus combining the data-driven design method normally associated with ANNs with the human-centred design techniques used to produce fuzzy systems. In section 6, a set of algorithms will be investigated that can automatically determine the structure of the rule base, but this section will consider the easier problem of determining the rule base directly from the data.

Assuming that the fuzzy variables that refer to the inputs and outputs, are all fixed and that COG defuzzification and singleton fuzzification algorithms are used together with algebraic fuzzy operators, then the output of the neurofuzzy network is given by equation 59, where the only unknown parameters are the *weights*, or equivalently the rule confidences. It can be shown that it is computationally much easier to adapt the weight vector and due to the invertible relationship that exists between a weight and its corresponding rule confidence vector, a fuzzy algorithm with linguistic outputs can be extracted from the trained neurofuzzy network

The network's output is linearly dependent on the weight vector, see equation 68, therefore any of the linear training rules are applicable to these networks. This means that the Least Mean Square and Normalised Least Mean Square instantaneous training algorithms can be used to cycle through the training set $\{\mathbf{x}^p, d^p\}_{p=1}^P$ and adapt the weights in an iterative manner. Similarly, the batch gradient descent and steepest descent algorithms can be used to train the parameters, where the whole training set is presented to the network *before* updating the parameters. However, because the output is a linear function of the weights, the optimal (with respect to the Mean Squared Error cost function) values can be calculated directly using the Moore-Penrose pseudo-inverse or using Singular Valued Decomposition algorithms.

6 Construction Algorithms

So far, this report has discussed how fuzzy and neurofuzzy systems can be implemented and also described how the weights can be updated using a variety of batch and instantaneous learning rules. This section looks at the basic structure of these networks: the number, form and position of the membership functions as well as the structure of the fuzzy rules, and describes some construction algorithms that can automatically learn an appropriate structure directly from a set of training data. It has been stressed that the main advantage of a fuzzy approach is that it can be used to encode prior expert knowledge, but a fuzzy representation is also useful for validating a network trained directly from a dataset. When an ANN or a neurofuzzy network is designed using a data-driven approach, the deficiencies that exist in the data are not always obvious to the designer and any network that is trained directly from input, output data should always be subject to a stringent set of verification and validation tests. These tests can be numerical, but other qualitative measures would include investigating how well the training data had excited the input space, how well the

network represented these regions and investigating the network's output surface by looking at smoothness features and also trying to validate any vague or symbolic rules that can be used to describe the system. Neurofuzzy systems have basis functions with a compact (or local) support and their behaviour can be described using a fuzzy algorithm, therefore they allow an expert to perform qualitative verification tests.

The construction techniques outlined in this section are biased towards the lattice-based B-spline neurofuzzy networks but similar techniques can also be applied to Gaussian RBF neurofuzzy systems.

6.1 Additive-type Decomposition

The curse of dimensionality occurs in neurofuzzy systems because of the fact that for a complete rule base, the number of rules is exponentially dependent on the dimension of the input space, see section 3.4.3. In order to simplify the structure of the neurofuzzy system and retain desirable properties such as rule-base completeness, the designer must understand how both the network and the desired function maybe structured. A convenient framework for this is the ANalysis Of VAriance (ANOVA) representation [10] which expresses an n-dimensional function as:

$$f(\mathbf{x}) = f_0 + \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n} \sum_{j=i+1}^{n} f_{i,j}(x_i, x_j) + \cdots + f_{1,2,\ldots,n}(\mathbf{x}) \tag{70}$$

where f_0 represents the function's bias and the other terms represent the univariate, bivariate etc. *additive* components of the function $f(.)$. It should immediately be noticed that each component in the ANOVA description can be approximated by a separate neurofuzzy system, as illustrated in figure 37. The ANOVA representation is simply a generalisation of the well-

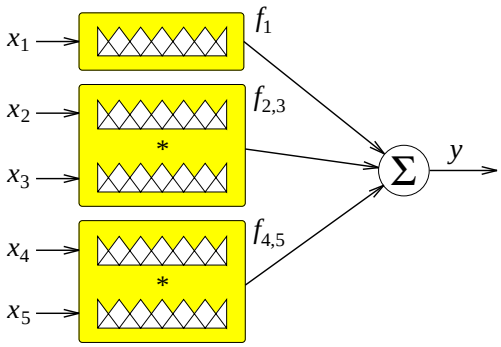


Figure 37: An additive ANOVA decomposition of a neurofuzzy rule base.

known Gabor-Kolmogorov polynomial expansion where each function is the product of its input components.

Any function can be represented in the ANOVA framework, and the advantage with this type of formalism is that it expands the unstructured mapping into simple and complex additive components, where each component can be modelled by a neurofuzzy network.

When a neurofuzzy network is used to approximate part of the ANOVA description, it is termed a *subnetwork*, so the output of the overall system is formed from the sum of various simpler subnetworks. A rule within each subnetwork may have the form:

$$\text{IF } (x_3 \textit{ is } PS \text{ AND } x_5 \textit{ is } AZ) \text{ THEN } (o \textit{ is } PS) \qquad 0.5 \tag{71}$$

where the antecedent of the rule involves only a small subset of the input vector, and the overall rule base looks something like:

	IF (x_1 *is* NM) THEN (o *is* PS)	1.0
OR	IF (x_2 *is* PS AND x_3 *is* AZ) THEN (o *is* PS)	0.7
OR	IF (x_4 *is* NS AND x_5 *is* PL) THEN (o *is* PM)	0.3

Each of the rule's antecedents is composed of only a small subset of the total number of input variables, hence the number of rules in that subnetwork is generally relatively small. It should be noted that this representation is only useful when a significant number of the ANOVA terms are identically zero, i.e. when the memory requirements for this representation is less than that used in a full neurofuzzy system. This is illustrated by the example shown in figure 37, where 7 basis functions are defined on each axis and the ANOVA system uses ≈ 100 weights whereas a full 5-dimensional neurofuzzy system would require $\approx 16,800$ weights, a potential saving of over 2 orders of magnitude.

Finally, it is worth emphasising that because each neurofuzzy subnetwork is *linear* in the weight vector (equivalently rule confidence matrix), the overall system is linear in the concatenated parameter vector. Hence, any of the basic convergence and stability results that are derived for standard neurofuzzy systems also apply to this ANOVA representation. This additive property also ensures that the form of output surface with respect to a particular input variable is the same as that of the appropriate subnetwork.

6.1.1 Rule Base Completeness

Letting $s_i(.)$ be the i^{th} neurofuzzy subnetwork in the ANOVA representation ($i = 1, \ldots, r$), and suppose each subnetwork forms a partition of unity, then the sum over all the subnetworks is r. Hence the overall neurofuzzy ANOVA network forms a partition of r. It is not difficult to incorporate this in the standard neurofuzzy theory as it is an arbitrary choice to make the membership functions and rule bases form a partition of unity; in fact it could be any constant value (although care would have to be taken over the choice of T and S-norms). Fuzzy and neurofuzzy systems could have membership values that ranged from 0 to 10, or from 0 to 0.1, as long as the information is treated consistently. Therefore, in an ANOVA neurofuzzy system, we should scale each subnetwork by a factor $1/r$ as this will make the overall ANOVA network form a partition of unity.

An ANOVA neurofuzzy network rule base is complete when all of the rule bases associated with each of the individual subnetworks are complete. In addition, when each input variable occurs at most once in the subnetworks, the ANOVA rule base is complete if and only if all the subnetworks' rule bases are complete. Therefore the global properties associated with the ANOVA expansion can be inferred from the properties of each of the individual subnetworks.

6.1.2 Basis Function Shape

Using an additive, ANOVA-type decomposition changes the generalisation properties of each of the basis (membership) functions, as they become *global* with respect to every input variable that does not contribute to that subnetwork. These basis functions have been termed *semi-local* as they're strictly local with respect to some input variables but do not depend on other measurements. They are characterised by the fact that their membership value is constant with respect to every input variable which is not part of that subnetwork, hence their contours are parallel to these axes. This is illustrated in figure 38, where a basis function is plotted that only depends on one variable (x_1).

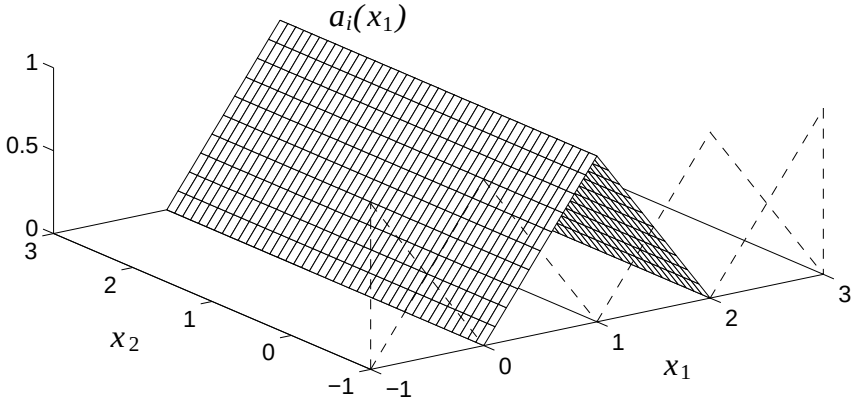


Figure 38: A semi-local basis function which only depends on x_1.

This distinction between local and semi-local basis functions is precisely why this ANOVA representation is useful. Often the training data is not distributed well enough to excite all of the basis functions on a lattice structure. However, using semi-local basis functions means that the network is able to locally extrapolate information parallel to a subnetwork's unmodelled input variables, thus potentially overcoming the curse of dimensionality.

6.1.3 Additive Functions

When all the bivariate and higher order terms in the ANOVA decomposition are identically zero, the function can be expressed as:

$$f(\mathbf{x}) = f_0 + \sum_{i=1}^{n} f_i(x_i) \tag{72}$$

This is called an *additive* function as it involves only *univariate* nonlinear functions, and it has been studied extensively for many years [14]. It is clear that this representation does not suffer from the curse of dimensionality as its computational cost grows *linearly* with the number of input variables. However, an additive network is unable to model functions which involve products of inputs and this can be a serious restriction.

An interesting iterative method for training these univariate functions which is loosely based on a block Gauss-Seidel technique has been developed called *backfitting*. Each subnetwork is iteratively trained to approximate the residual signal (the difference between the desired function and the current network's output) so each subnetwork tends closer to its optimal value [26].

6.2 ANOVA Parameterisation

The basic ANOVA framework is very appealing as it retains all of the important properties of a neurofuzzy network (smoothness, linear in the parameter vector, simple fuzzy interpretation etc.) and yet has the ability to reduce the effect of the curse of dimensionality when the unknown function is structured appropriately. However, it is difficult for human designers to correctly articulate the structure of the subnetworks (which inputs are important, how many basis functions, how many subnetworks?), therefore it is appropriate to investigate how this task would benefit from a data-driven approach where these parameters are determined directly from the training data.

Building an ANOVA-type neurofuzzy system directly from a set of training data therefore involves determining:

- whether or not an input measurement is relevant for the current modelling problem and if so, how it interacts with the other important variables, and
- the number of basis functions in each subnetwork as well as calculating their position.

6.2.1 Input Variable Selection

One important consideration for developing parsimonious neurofuzzy networks is to determine which input components are useful in predicting the value of the unknown function. Each redundant input variable increases the network's complexity without adding any useful flexibility. Unfortunately, determining which inputs (or combinations of inputs) are significant requires an extensive search procedure as the number of possible combinations of subnetworks with r inputs where the original input space is n-dimensional is equal to:

$$^{n}C_{r} = \frac{n!}{r!(n-r)!} \tag{73}$$

and this is illustrated in table 2 for various values of n and r. As can be seen from this table,

	n						
r	1	2	3	5	8	13	21
1	1	2	3	5	8	13	21
2		1	3	10	28	78	210
3			1	10	56	286	1,330
5				1	56	1,287	20,349
8					1	1,287	203,490

Table 2: The number of ways of selecting r from n inputs variables. The blank entries denote impossible situations.

when n and r are large, an unfeasibly large search procedure must be carried to determine which additive subnetwork structure is appropriate, therefore methods must be developed to determine which subnetwork structures are relevant. These are generally based on one-step-ahead inductive learning-type procedures, as will be described in section 6.3.

6.2.2 Basis Function Selection

Choosing which basis (fuzzy membership) function distribution is appropriate for a particular variable is a complex, nonlinear optimisation problem, which involves introducing and deleting certain basis functions as well as altering the shape of the remaining ones. The techniques used to implement these functions are very much dependent on the type of basis functions used in the neurofuzzy network, and here we will be exclusively concerned with B-spline basis functions.

Univariate B-spline basis functions are piecewise polynomials of order k defined on a set of intervals which partition of the input variable. On each interval, the function is a polynomial or order k and the partition values are contained in the knot vector, see section 5.1.1. Introducing a new basis function amounts to inserting a new interior knot in the knot vector and deleting a basis function is equivalent to removing a knot, as illustrated in figure 39.

Introducing a knot means that the new network can reproduce the old one exactly (because they're both polynomials across the old interval), but the new network also has an extra degree of flexibility at the new knot because the surface can now be piecewise linear. Deleting a knot is equivalent to removing this degree of flexibility and two intervals are merged into one. As the nonlinear optimisation procedure for placing these knots is very hard (badly conditioned),

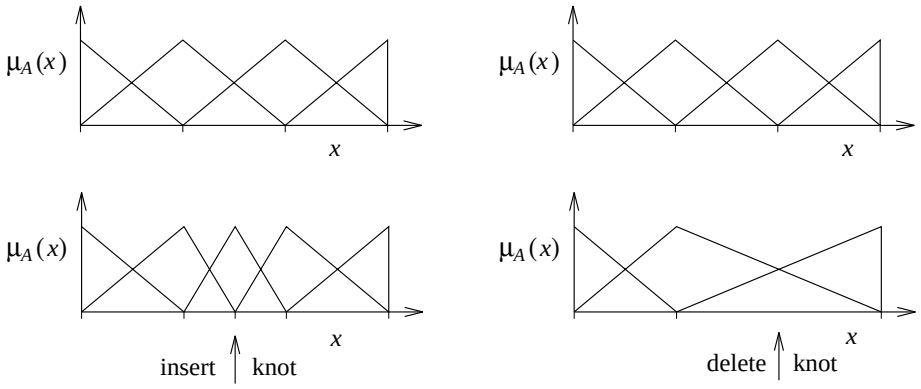


Figure 39: Adding and removing a basis function is equivalent to introducing a new knot and deleting an old one, respectively.

it is usual to just consider knot insertion and deletion, where new knots can be inserted at the centre of an interval, rather than adapting their values directly. This is a widely used technique in inductive learning algorithms, and it generally simplifies the procedure.

It is worth commenting on the number of new basis functions inserted in a multivariable subnetwork, when one of the fuzzy variables has a knot inserted. Assuming that the subnetwork has r inputs, and there exist p_j basis functions on each axis ($j = 1, \ldots, r$), then the number of new multivariable basis functions introduced by inserting a knot on the i^{th} axis is:

$$\Delta p = \prod_{\substack{j=1 \\ j \neq i}}^{r} p_j \tag{74}$$

This occurs because the linguistic intersection (AND) must be taken between the new univariate basis function and all the remaining basis functions defined on the other axes, in order to retain the partition of unity property.

6.3 ANOVA Construction Algorithms

One important iterative construction algorithm for B-spline neurofuzzy systems was initially developed and proposed by Kavli in 1992, [19], which attempts to construct an additive neurofuzzy ANOVA model of the unknown function. The Adaptive Spline Modelling of Observation Data (ASMOD) procedure works by a one step ahead error minimisation procedure, as at each time step a number of possible rule base refinements are evaluated and the one that is most statistically significant is incorporated in the neurofuzzy system.

Given a current ANOVA neurofuzzy model (possibly empty), possible *candidate* refinements that extend the flexibility of the model are to:

1. include extra inputs in the model by introducing a new additive univariate subnetwork,
2. extend the flexibility of a subnetwork by introducing a new variable in it (taking the tensor product of the old subnetwork with a univariate subnetwork), and therefore increasing its dimension by one, and
3. introducing a new basis function in a subnetwork.

Each type of refinement extends the flexibility of the network in different ways and they all introduce different numbers of new parameters into the network. Obviously, a more flexible network should be able to fit the data better, but these construction algorithms are aimed at

producing *parsimonious* systems [5], as a simpler network is both easier to verify and validate and cheaper to implement. Therefore instead of of just choosing the refinement that reduces the MSE the most, it is usual to weight this reduction with factors that take into account the number of parameters introduced into the problem. Probably the most common such measure is the Baysian Statistical Significance (BSS) measure given by:

$$E_{BSS} = P \ln (E_{MSE}) + p \ln P \tag{75}$$

where J_{BSS} and J_{MSE} are the BSS and MSE performance functions, respectively, p is the number of parameters in the model and L is the amount of training data. The refinement that is the most statistically significant is included in the current model and this procedure is then repeated until the overall model is sufficiently accurate. This process is illustrated in figure 40.

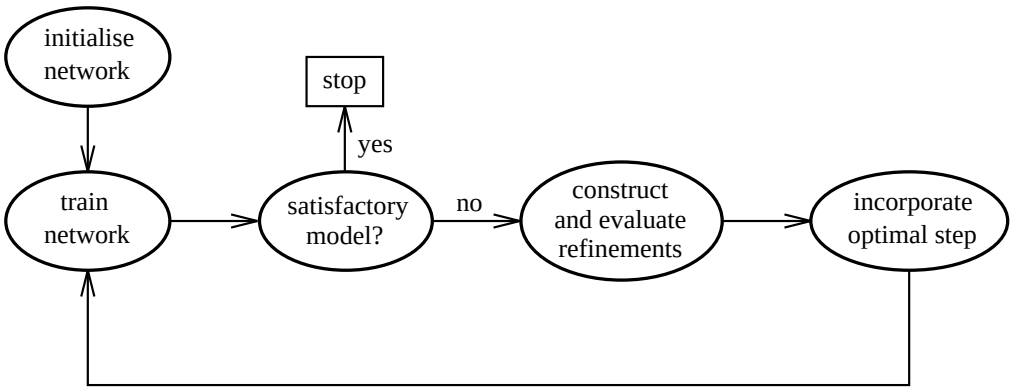


Figure 40: A typical one-step-ahead inductive learning algorithm cycle, as used in the ASMOD algorithm.

Although the ASMOD procedure was originally proposed as a constructive algorithm, it is both desirable and necessary to introduce a set of pruning refinements which simplify the network's structure. This is because either the algorithm may be initialised with a complex rule base and the designer would like to see whether there exists a simpler, more parsimonious network that can form an adequate model, or else to "backtrack" when the construction algorithm makes an inappropriate refinement. The three possible pruning actions are the mirror of the candidate refinements:

1. remove a univariate, additive subnetwork,
2. split up a tensor product subnetwork into several simpler subnetworks, and
3. remove a set of basis functions from a subnetwork by deleting a knot.

6.3.1 Limitations and Possible Solutions

The ASMOD algorithm is a one-step-ahead iterative construction procedure, and like all such techniques it can have problems where a single refinement does not indicate any improvement but a k-step-refinement would vastly improve the network's structure. As an example of this, consider the following function:

$$f(x_1, x_2) = \sin(\pi\, x_1 x_2)$$

[5] A network should be flexible enough to model the data to a specified accuracy, but no larger.

where the training samples are drawn with a uniform probability from the input space $[-1,1]\times[-1,1]$. The integral with respect to either x_1 or x_2 (due to symmetry) is identically zero, so including an additive term in either x_1 or x_2 would not result in a reduction in the error and a one-step-ahead construction algorithm would never form a bivariate function. This situation could be overcome by looking more than one-step-ahead but (not surprisingly) the number of possible refinements increases almost exponentially fast, and some "intelligence" must be used in evaluating which ones to consider. This is similar to the heuristics developed to search large trees in a near-optimal fashion.

6.4 ASMOD Example

The ASMOD structuring algorithm is now used to construct a suitable B-spline model for predicting the two-input, single-output nonlinear mapping described by:

$$\begin{aligned} o(t) &= \left(0.8 - 0.5\exp\left(-o^2(t-1)\right)\right) o(t-1) - \\ &\quad \left(0.3 + 0.9\exp\left(-o^2(t-1)\right)\right) o(t-2) + 0.1\sin(\pi o(t-1)) + \eta(t) \end{aligned} \tag{76}$$

where $\eta(t)$ denotes the additive noise at time t. When $\eta(t) \equiv 0$, this difference equation has an unstable equilibrium at the origin and a globally attracting limit cycle as shown in figure 41. The "surface" of the time series which the above difference equation represents is

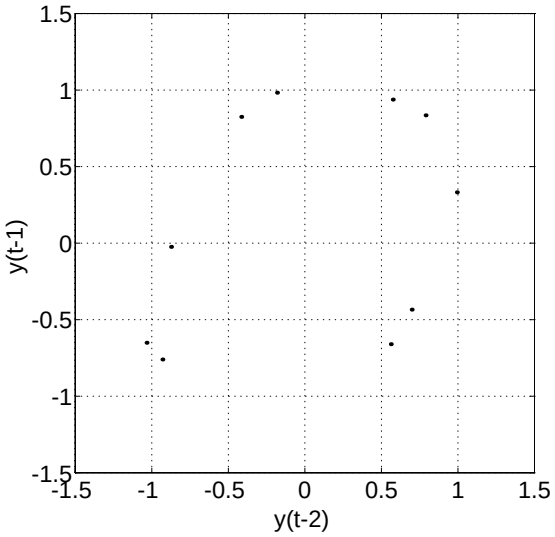


Figure 41: Iterated time series mapping from the initial condition $(0.1, 0.1)^T$. The time series "spirals" out from the unstable equilibrium at the origin towards the globally attracting limit cycle.

also shown in figure 42. From this figure and equation 76, it can clearly be seen that when $o(t-1)$ is held constant, $o(t)$ is *linear* in $o(t-2)$ and so the basic structure of the desired function with respect to the two inputs is different.

Defining the two-dimensional input vector $\mathbf{x}(t) = (o(t-1), o(t-2))^T$ and using $o(t)$ as the (scalar) desired output, a set of noisy training samples can be collected by iterating equation 76 from an initial condition $\mathbf{x}(1)$. The data set then represents the noisy dynamical behaviour of the discrete time series. From an initial condition $\mathbf{x}(1) = (0,0)$, the 300 training inputs that represent the noisy iterated dynamics of equation 76 are shown as a scatter plot in figure 43.

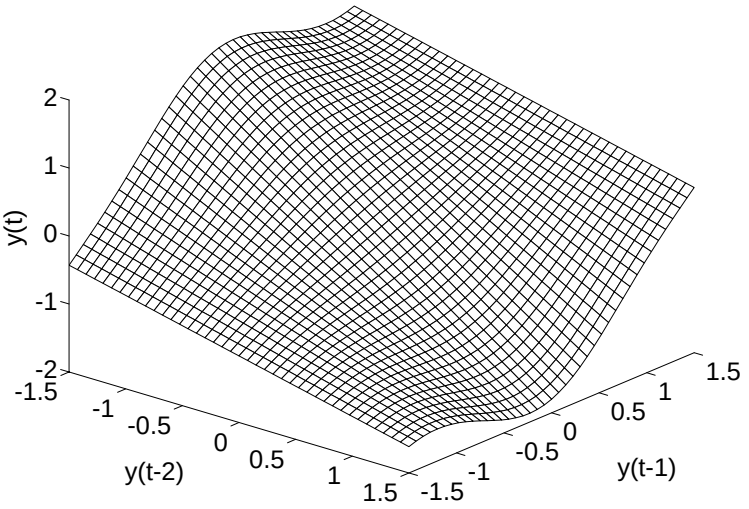


Figure 42: Time series output surface $o(t)$ v. $o(t-1) \times o(t-2)$, where $y(t)$ lies in the interval $[-1.872, 1.872]$.

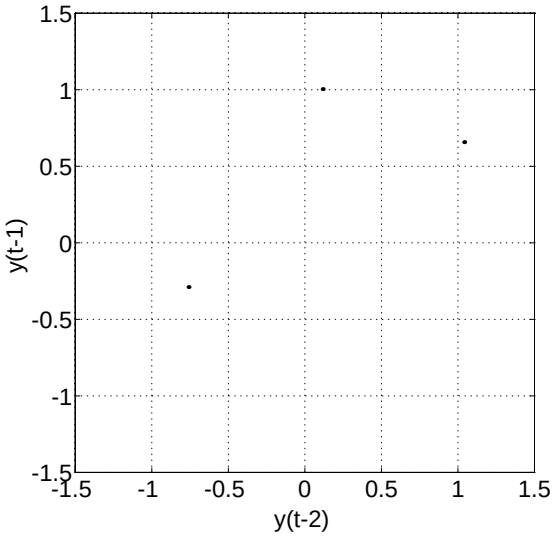


Figure 43: The noisy iterated dynamics of the time series from an initial position at the origin.

The unknown recurrence relationship can also be expressed as:

$$o(t) = f(o(t-1)) + g(o(t-1))o(t-2) + \eta(t) \tag{77}$$

for two appropriate nonlinear functions $f(.)$ and $g(.)$, and an additive disturbance $\eta(t)$. The aim of this section is to illustrate how the ASMOD procedure works and to demonstrate that incorporating *a priori* knowledge into the B-spline network design improves its generalisation abilities. Neural networks are often cited as being model free estimators, but this is generally not true. The network's structure is generally decided *a priori* by the designer and a set of parameters (weights) are adapted using an appropriate learning algorithm. Due to the underlying flexibility of many of the networks considered in this report, the authors have termed these algorithms *weak* or *soft* modelling algorithms. One of the main advantages of using B-spline networks is that they allow limited *a priori* knowledge to be incorporated into their structure, although it should be remembered that if the underlying structure of the network is not flexible enough (or even too flexible) to represent the training data adequately (a biased system), it performs poorly.

6.4.1 ASMOD Refinements

Equation 77 shows how the discrete recurrence relationship can be expressed as a sum and product of two nonlinear *univariate* functions $f(.)$ and $g(.)$. The important point to notice is that the desired function is a linear function of $o(t-2)$, when $o(t-1)$ is held constant, whereas it is a strongly nonlinear function of $o(t-1)$. If this type of knowledge is available during the network design process, it should be incorporated into the basic network structure, as it simplifies the learning process. Otherwise any algorithm that adapts the network's structure should be able to generate such information automatically. This kind of *a priori* knowledge can easily be incorporated into the B-spline network design by defining univariate basis functions of different shapes and sizes (orders) on each (univariate) axis, as the multivariate basis functions are formed by taking the tensor product.

To begin the ASMOD procedure, two, order 2 univariate basis functions were used to represent both $o(t-1)$ and $o(t-2)$ on the input domain $[-1.5, 1.5]$. The initial optimal ASMOD network was empty, so one of these univariate submodels must be included from the store to start the refinement algorithm. The most statistically significant subnetwork s_2 to include was the one representing $o(t-2)$ which produced the model output shown in figure 44. After this refinement, the network was able to model the variable $o(t-2)$ additively.

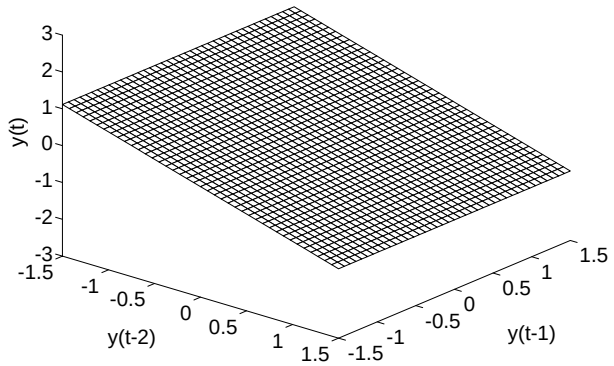


Figure 44: The first ASMOD refinement which introduces a univariate subnetwork on $o(t-2)$. The network's output lies in the range $[-1.12, 1.12]$.

Three possible candidate refinement steps could then be performed: inserting a knot at zero

in s_2, introducing the univariate function $s_1(o(t-1))$ as another subnetwork or forming a tensor product between s_2 and s_1 resulting in a new B-spline subnetwork $s_3(o(t-1), o(t-2))$. The tensor product refinement was estimated to be the best, and the new output surface is

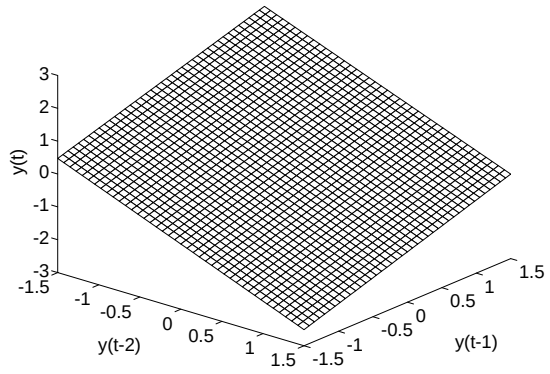


Figure 45: The second ASMOD refinement step which forms a tensor produce model. The network's output lies in the range $[-2.45, 2.51]$.

shown in figure 45. Only two possible actions could now be taken: a knot insertion at zero on either axis. Inserting a knot at zero on $o(t-1)$ was statistically the most significant and any further refinements reduced the amount of information stored in the network, according to the BSS statistical significance measures. Therefore the final model was composed of just one subnetwork $s_3(.)$, which has two piecewise linear basis functions defined on $o(t-2)$ and three on $o(t-1)$, resulting in the prediction surface shown in figure 46.

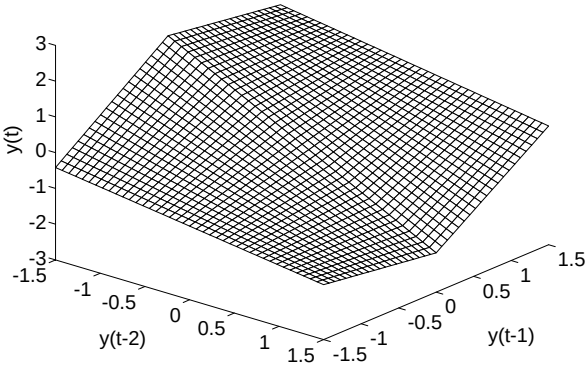


Figure 46: The final ASMOD model which has had a knot inserted at $o(t-1) = 0$. The network's output lies in the range $[-1.90, 1.94]$.

It is important to notice that the ASMOD algorithm has discovered the correct representation for $o(t-2)$ and models it using a globally linear mapping.

6.4.2 Model Evaluation

The network's output is plotted in figure 46, and while this may seem a gross simplification of the original surface, it is sufficient for representing the dynamical behaviour. This is confirmed in figure 47 where the iterated dynamics of the B-spline network is displayed. It is a very good approximation to the true plot shown in figure 41, as the knowledge that has been incorporated into the network's structure improves its ability to generalise in the interior of the limit cycle. It also improves the network's ability to extrapolate information,

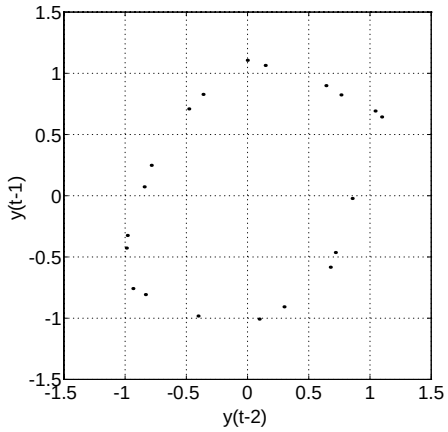


Figure 47: Iterated dynamics of a B-spline network from the initial condition $\mathbf{x}(1) = (0.1, 0.1)^T$. The very good approximation of the interior five arm spiral is because of the knowledge encoded in the model's structure.

with the minimum and maximum network outputs being -1.83 and 1.89, respectively, which are very close to the true values.

The network's prediction ability is shown in figure 48 where the mean square k-step-ahead prediction errors are plotted. It shows that good estimates can be obtained by iterating the

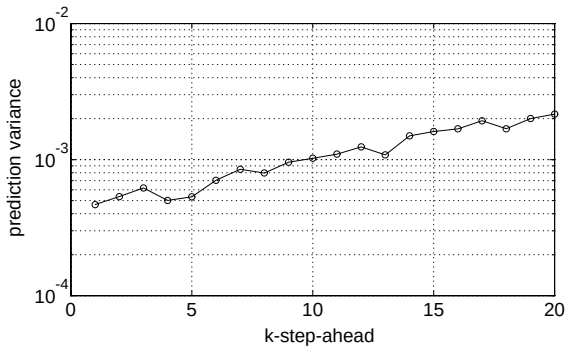


Figure 48: Variance of the k-step-ahead prediction errors.

network up to (at least) 20 steps into the future and this is confirmed in figure 49, where the network and the noiseless time series are run in parallel for 50 iterations. As can be seen from these two figures, there is very good agreement between the true and the estimated time history, and other statistical tests confirm that this is a good, simple model of the "unknown function".

6.5 Alternative Input Space Partitioning Algorithms

The lattice partitioning of the original input space that is generated by many fuzzy and neurofuzzy systems is convenient because the linguistic interpretation is easy to formulate as the logical intersection (AND) of the individual univariate fuzzy sets and it is computationally efficient to implement. However, there is sometimes conflict between modelling and representational requirements as a particular input space partitioning scheme may not have a direct fuzzy interpretation, despite it being the most appropriate for the problem under consideration. This is not really a serious objection though, as any axis orthogonal partition

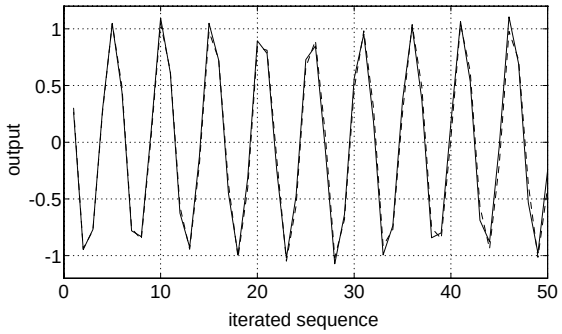


Figure 49: The noiseless time series and the B-spline network iterated mappings from an initial condition $o(-1) = 0.5$, $o(0) = 1.0$. The dashed line represents the true time series and the solid line shows the B-spline iterated mapping.

can be represented as a fine lattice with a set of constraints determining the outputs of the extra basis functions; a kind of *regularisation*[6]. There is also a direct interpretation of the fine lattice as a (larger) set of fuzzy rules and these could be used to aid the designer, but the basic principle of developing a parsimonious rule base would not apply.

The aim of abandoning the lattice based representation of a standard neurofuzzy system is to try to achieve a parsimonious representation within each *subnetwork*. It is still desirable to model the ANOVA additive relationships, but each lattice-based subnetwork also suffers from the curse of dimensionality. Therefore consider the two alternative schemes shown in figure 50. Both of these schemes have the potential to model local variations with fewer

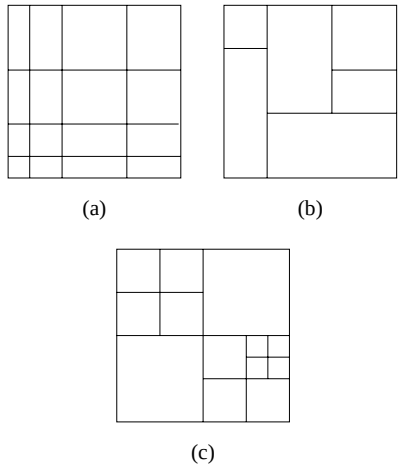


Figure 50: Three different input space partitioning strategies: (a) a lattice (b) a k-tree and (c) a quad tree.

basis functions as a k-tree only introduces 1 new basis function when a knot is added and a quad-tree introduces 2^n, both of which are generally significantly smaller than that given in equation 74, [3].

All of these methods produce axis orthogonal splits, which while aiding the representational aspects of the neurofuzzy system (see the comments made about Gaussian neurofuzzy systems in section 5.1.2), can limit the modelling abilities of the subnetwork (although pro-

[6]Regularisation is a technique which incorporates an extra term in the cost function which measures the *curvature* of the output surface, thus biasing the network to produce smooth output surfaces.

jection pursuit may help, see the next section), and examples using both these schemes have appeared in the literature.

6.5.1 Hierarchical Systems

k-trees and quad-trees can also be imagined as providing a hierarchical-type rule base where an area is recursively (hierarchically) decomposed until the discretisation is fine enough to allow an adequate approximation, as illustrated in figure 51 However, hierarchical fuzzy and

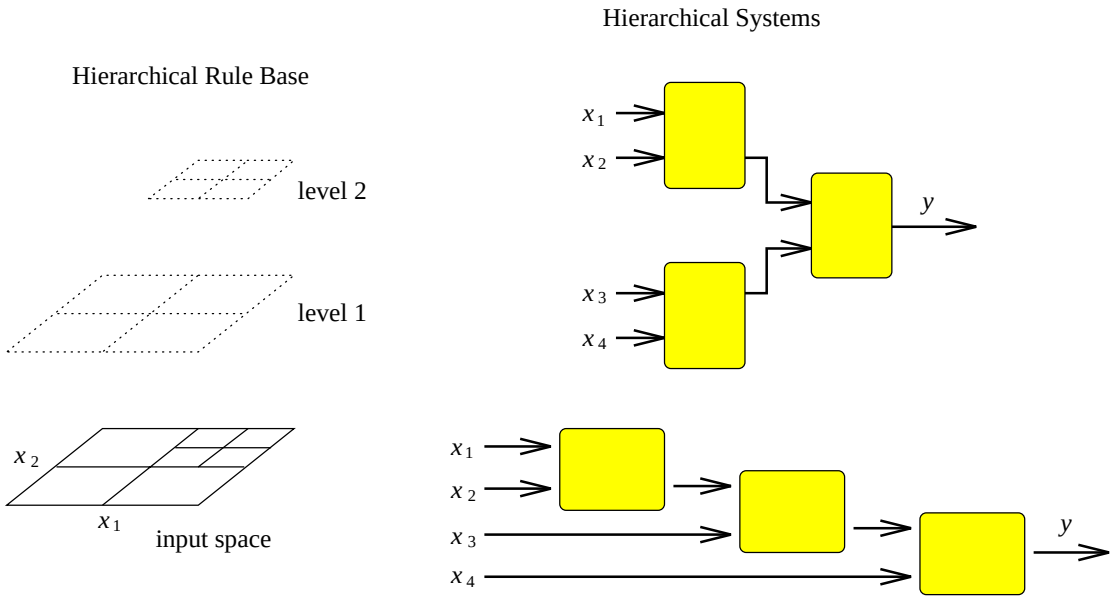


Figure 51: A comparison between a fuzzy hierarchical rule base and two hierarchical systems.

neurofuzzy systems are usually composed of several subnetworks (like the ANOVA representation) but they are arranged to form deep, nested concepts where the output of one network is an input to another. This is also illustrated in figure 51, and it can clearly be seen that the overall aim is to produce parsimonious representations by structuring the overall network from smaller dimensional subnetworks. These structures however, do not retain the additive property of the ANOVA representation and as such require complex nonlinear optimisation algorithms even to train the weight vector in each subnetwork.

All of these more advanced neurofuzzy systems retain the partition of unity property within each subnetwork, although, apart from the ANOVA representation, the fuzzy algorithm's structure is more complex and unless the application demands it, the simpler neurofuzzy ANOVA networks is a powerful methodology that is fairly simple to apply.

7 Summary

Fuzzy sets and fuzzy logic provide a means for representing an expert's knowledge on a computer, where the graded representation of a fuzzy set allows a gradual transition from one rule to another, without the hard breaks or jumps that are due to using standard binary sets. This may appear to be a fairly simplistic generalisation of conventional rule-based system, but probably the main result of the hundreds (possibly thousands) of applications developed in the Far East is that this representation is natural for both the designer and the expert and that the restricted natural language term set employed in fuzzy logic is rich enough to represent just about all of the ideas necessary for developing real-world systems. Indeed,

a fuzzy system uses a linguistic interface which is much closer to the problem domain than to its actual implementation, and this makes it easier for the expert to validate and create new behaviours within the system. Researchers and engineers should not be blinded by the data-driven design technologies (such as ANNs) as sometimes it is difficult to obtain training data, but an expert may have a good, high-level understanding of the basic processes. Techniques such as fuzzy logic can make use of this information source and indeed the neurofuzzy systems can be "trained" using both numerical and expert information.

This report provides an incomplete survey of the field of fuzzy logic. Notable omissions from the description include the fuzzy c-means clustering algorithm [4], discrete fuzzy systems and the development of self-organising controllers [29], rough set theory and several other important areas. However, its aim is to provide the reader with the basic concepts involved: fuzzy sets and membership functions, fuzzy operators and rule bases as well as to describe the relationship that exists between fuzzy logic and artificial neural networks: the so called *neurofuzzy systems*. This has been achieved by focusing on B-spline and Gaussian neurofuzzy systems and it has been argued that the type of fuzzy operators that are used in these networks are also appropriate for most fuzzy systems as the quality of the decision surface is improved. Concepts that are useful for neurofuzzy systems, such as data excitation, regularisation, construction algorithms have their parallels in fuzzy networks and the extra insights that they provide are useful for validating the trained network.

This report has also highlighted the comparatively simple structure of a neurofuzzy system and shown how it can be trained using relatively simple *linear* optimisation techniques. The problems associated with the curse of dimensionality have been outlined and some techniques for overcoming it have been described. The report's aim therefore has been to describe the basic principles associated with fuzzy and neurofuzzy systems as well as outlining how the simplest structures may be developed in practice.

References

[1] J.F. Baldwin, T.P. Martin, and Pilsworth B.W. *FRIL - Fuzzy Evidential Reasoning in AI.* Research Studies Press (Wiley), 1995.

[2] R.E. Bellman. *Adaptive Control Processes.* Princeton University Press, Princeton, NJ, 1961.

[3] C.S. Berger. Linear splines with adaptive mesh sizes for modelling nonlinear systems. *IEE Proc. Part D*, 141(5):277–284, 1994.

[4] J.C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Plenum Press, New York, 1981.

[5] M. Brown and C. Harris. *Neurofuzzy Adaptive Modelling and Control.* Prentice Hall, Hemel-Hempstead, UK, 1994.

[6] M. Brown and C.J. Harris. A nonlinear adaptive controller: A comparison between fuzzy logic control and neurocontrol. *IMA J. Math. Control and Info.*, 8(3):239–265, 1991.

[7] E. Cox. *The Fuzzy Systems Handbook.* AP Professional, 1994.

[8] M.G. Cox. Algorithms for spline curves and surfaces. NPL Report DITC 166/90, 36 pages, 1990.

[9] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control.* Springer-Verlag, Berlin, 1993.

[10] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–141, 1991.

[11] J.H. Friedman and W. Stuetzle. Projection pursuit regression. *J. American Statistical Association*, 76(376):817–823, 1981.

[12] C.J. Harris, editor. *Advances in Intelligent Control.* Taylor and Francis, London, 1994.

[13] C.J. Harris, C.G. Moore, and M. Brown. *Intelligent Control: Some Aspects of Fuzzy Logic and Neural Networks.* World Scientific Press, London & Singapore, 1993.

[14] T.J. Hastie and R.J. Tibshirani. *Generalized Additive Models*, volume 43 of *Monographs on Statistics and Applied Probability.* Chapman & Hall, 1990.

[15] P.J. Huber. Projection pursuit. *The Annals of Statistics*, 13(2):435–475, 1985.

[16] K.J. Hunt, R. Haas, and M. Brown. On the functional equivalence of fuzzy inference systems and spline-based networks. *Int. J. Neural Systems*, 6(2):171–184, 1995.

[17] J.N. Hwang, S.R. Lay, M. Maechler, D.R. Martin, and J. Schimert. Regression modeling in back-propagation and projection pursuit learning. *IEEE Trans. on Neural Networks*, 5(3):342–353, 1994.

[18] J.S.R. Jang and C.T. Sun. Neuro-fuzzy modeling and control. *Proc. IEEE*, 83(3):378–406, 1995.

[19] T. Kavli. ASMOD - an algorithm for Adaptive Spline Modelling of Observation Data. *Int. J. Control*, 58(4):947–968, 1993.

[20] B. Kosko. Fuzziness vs probability. *Int. J. General Systems*, 17(2):211–240, 1990.

[21] B. Kosko. *Neural Networks and Fuzzy Systems.* Prentice-Hall, Englewood Cliffs, NJ, 1992.

[22] B. Kosko. *Fuzzy Thinking: The New Science of Fuzzy Logic.* Harper and Collins, 1994.

[23] S.H. Lane, D.A. Handelman, and J.J. Gelfand. Theory and development of higher order cmac neural networks. *IEEE Control Systems Mag.*, pages 23–30, April 1992.

[24] M. Laviolette and J.W. Seaman. The efficacy of fuzzy representations of uncertainty. *IEEE Trans. on Fuzzy Systems*, 2(1):4–15, 1994. Discussion articles on pp. 16-42.

[25] D. McNeill and P. Freiberger. *Fuzzy Logic.* Simon & Schuster, New York, NY, 1993.

[26] D.J. Mills, M. Brown, and C.J. Harris. Training neurofuzzy systems. In *Proc. IFAC Symposium on AI in Real-Time Control*, pages 213–218, Valencia, 1994.

[27] R. Murray-Smith. *A local model network approach to nonlinear modelling.* PhD thesis, Department of Computer Science, University of Strathclyde, 1994.

[28] W. Pedrycz. *Fuzzy Control and Fuzzy Systems.* Research Studies Press, John Wiley and Sons, Taunton, 2^{nd} edition, 1993.

[29] T.J. Procyk and E.H. Mamdani. A linguistic self-organising process controller. *Automatica*, 15:15–30, 1979.

[30] J.L.A. van Rijckevorsel. Fuzzy coding and b-splines. In J.L.A. van Rijckevorsel and J. de Leeuw, editors, *Component and Correspondence Analysis*, pages 33–54, Chichester, 1988. John Wiley and Sons. chapter 2.

[31] B. Russell. Vagueness. *Australian J. Philosophy*, 1, 1923.

[32] J.R. Shewchuk. An introduction to the conjugate gradients method without the agonizing pain. Technical Report CMU-CS-94-125, School of Computer Science, Carnegie Mellon University, 1994.

[33] R. Sutton and I.M. Jess. A design study of a self-organizing fuzzy autopilot for ship control. *IMechE, Proc. Instn. Mech. Engrs.*, 205:35–47, 1991.

[34] L.X. Wang. *Adaptive Fuzzy Systems and Control: Design and Stability Analysis.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[35] H.W. Werntges. Partitions of unity improve neural function approximation. In *Proc. IEEE Int. Conf. Neural Networks*, pages 914–918, San Francisco, CA, 1993. Vol. 2.

[36] B. Widrow and M.A. Lehr. 30 years of adaptive neural networks: Perceptron, madaline and backpropagation. *Proc. IEEE*, 78(9):1415–1441, 1990.

[37] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

[38] L.A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. on System Man and Cybernetics*, 3(1):28–44, 1973.

[39] H.J. Zimmermann. *Fuzzy Set Theory and its Applications.* Kluwer Academic Press, Boston, MA, 2nd edition, 1993.