IMAT5119 Fuzzy Logic – Theoretical Assignment

Jon-Paul Boyd
School of Computer Science and Informatics
De Montfort University
United Kingdom

Question 1 – Fuzzy Sets

A fuzzy set A in X (classical set of objects, called the universe, whose generic elements are denoted x) is a set of ordered pairs:

$$A = \{(x, \mu_A(x)) | x \in X, \mu_A(x) \in [0,1]\}.$$

The $\alpha - cut$ ($\alpha \in [0,1]$) of a fuzzy set A is the ordinary set

$$A_\alpha = \{x \in X | \mu_\alpha(x) \geq \alpha\}.$$

The $strong\ \alpha - cut$ ($\alpha \in [0,1]$) of a fuzzy set A is the ordinary set

$$A_{\alpha^+} = \{x \in X | \mu_\alpha(x) > \alpha\}.$$

The set of all distinct numbers in [0, 1] that are employed as membership grades of the elements of X in A is called the *level set* of A, denoted by $L(A)$. Assume minimum and maximum operators for the intersection and union of fuzzy sets. Answer the following:

Question 1a

Given any two fuzzy sets A and B, prove that the following properties hold:

$$(A \cup B)_\alpha = A_\alpha \cup B_\alpha \quad and \quad (A \cap B)_\alpha = A_\alpha \cap B_\alpha.$$ **(8 marks)**

Answer 1a

$$\text{Let } A = \{0.2/x_1 + 0.3/x_2 + 0.4/x_3 + 0.7/x_4 + 0.1/x_5\}$$

$$\text{Let B} = \{0.4/x_1 + 0.5/x_2 + 0.6/x_3 + 0.8/x_4 + 0.9/x_5\}$$

$$\text{Let } \alpha = 0.6$$

For the conjunction

$$(A \cup B)_\alpha = A_\alpha \cup B_\alpha$$

For the left-hand side (lhs) using the membership function using the max operator

$$(A\ U\ B) = \max[\mu A(x), \mu B(x)]$$

$$(A\ U\ B) = \{0.4/x_1 + 0.5/x_2 + 0.6/x_3 + 0.8/x_4 + 0.9/x_5\}$$

Accordingly

$$(A \cup B)_{0.6} = \{x_3, x_4, x_5\}$$

And for the right-hand side (rhs)

$$A_{0.6} = \{x_4\}$$

$$B_{0.6} = \{x_3, x_4, x_5\}$$

Hence

$$A_{0.6} \cup B_{0.6} = \{x_3, x_4, x_5\}$$

Therefore

$$(A \cup B)_\alpha = A_\alpha \cup B_\alpha$$

Similarly, for the intersection

$$(A \ \cap B)_\alpha = A_\alpha \cap B_\alpha$$

For the lhs using the membership function using the min operator

$$(A \ \cap \ B) = \min[\mu A(x), \mu B(x)]$$

$$(A \ \cap \ B) = \ \{0.2/x_1 + 0.3/x_2 + 0.4/x_3 + 0.7/x_4 + 0.1/x_5\}$$

Accordingly

$$(A \cap B)_{0.6} = \{x_4\}$$

And for the rhs

$$A_{0.6} = \{x_4\}$$

$$B_{0.6} = \{x_3, x_4, x_5\}$$

Hence

$$A_{0.6} \cap \ B_{0.6} \ = \{x_4\}$$

Therefore

$$(A \ \cap B)_\alpha = A_\alpha \cap B_\alpha$$

Proving the LHS and RHS in both the conjunction and intersection are commutative.

Question 1b

The support of A, denoted $supp(A)$, is defined as the set of elements of X that have nonzero membership in A. The core of A, denoted $core(A)$, is defined as the set of elements of X that have membership in A equals to 1.

How do $supp(A)$ and $core(A)$ relate to the $\alpha - cuts$ and the $strong\ \alpha - cuts$ of A? **(5 marks)**

Answer 1b

$$\text{Let } A = \{0.2/x_1 + 0.3/x_2 + 0.4/x_3 + 0/x_4 + 1/x_5\}$$

$$\text{Let } \alpha = 0$$

Considering the support of A is the crisp set with non-zero membership, hence

$$\text{supp}(A) = \{x_1, x_2, x_3, x_5\}$$

And the strong α-cut includes membership grades from A greater than α, so

$$A_{0^+} = \{x_1, x_2, x_3, x_5\}$$

Consequently

$$\text{supp}(A) = A_{0^+}$$

Considering the core of A is the crisp set with membership grade 1

$$\text{core}(A) = \{x_5\}$$

Now

$$\text{Let } \alpha = 1$$

So

$$A_1 = \{x_5\}$$

Therefore

$$\text{core}(A) = A_1$$

Question 1c
Given the discrete fuzzy sets $A = \{(0.2, x_1), (0.4, x_2), (0.6, x_3), (0.8, x_4), (1, x_5)\}$, obtain $L(A)$, and provide all the distinct $\alpha - cut$s of A.

What is the relationship between A_{α_1} and A_{α_2} when $\alpha_1 < \alpha_2$? **(7 marks)**

Answer 1c
The level set of A is the distinct membership grades of x in A, therefore

$$\mathrm{L}(A) = \{0.2, 0.4, 0.6, 0.8, 1\}$$

With the distinct $\alpha - cut$s as follows

$$A_{0.2} = 1/x_1 + 1/x_2 + 1/x_3 + 1/x_4 + 1/x_5$$

$$A_{0.4} = 0/x_1 + 1/x_2 + 1/x_3 + 1/x_4 + 1/x_5$$

$$A_{0.6} = 0/x_1 + 0/x_2 + 1/x_3 + 1/x_4 + 1/x_5$$

$$A_{0.8} = 0/x_1 + 0/x_2 + 0/x_3 + 1/x_4 + 1/x_5$$

$$A_1 = 0/x_1 + 0/x_2 + 0/x_3 + 0/x_4 + 1/x_5$$

With regard to the relationship between A_{α_1} and A_{α_2} when $\alpha_1 < \alpha_2$, this implies that $A_{\alpha_1} \supseteq A_{\alpha_2}$ meaning that $A_{\alpha_1}$is a superset of (includes) A_{α_2}

Question 1d
The membership function of A can be expressed in terms of the characteristic functions of its $\alpha - cuts$ according to the formula:

$$\mu_A = \sup_{\alpha \in [0,1]} \alpha \cdot \mu_{A_\alpha}(x) \text{ where } \mu_{A_\alpha}(x) = \begin{cases} 1 & iff\ x \in A\_\alpha \\ 0 & otherwise \end{cases}$$

In the case of a discrete fuzzy set we have $\alpha \in L(A)$. Show that this is true for the discrete fuzzy set given in c. **(5 marks)**

Answer 1d
Consider the discrete fuzzy set $A = \{(0.2, x_1), (0.4, x_2), (0.6, x_3), (0.8, x_4), (1, x_5)\}$

The level set for A is $\mathrm{L}(A) = \{0.2, 0.4, 0.6, 0.8, 1\}$

The alpha cut characteristic function can be determined with the following analytical form of the membership function

$$\mu_{A_\alpha}(x) = \begin{cases} 1 & iff\ x \in A\_\alpha \\ 0 & otherwise \end{cases}$$

Giving the following distinct $\alpha - cuts$

$$A_{0.2} = 1/x_1 + 1/x_2 + 1/x_3 + 1/x_4 + 1/x_5$$

$$A_{0.4} = 0/x_1 + 1/x_2 + 1/x_3 + 1/x_4 + 1/x_5$$

$$A_{0.6} = 0/x_1 + 0/x_2 + 1/x_3 + 1/x_4 + 1/x_5$$

$$A_{0.8} = 0/x_1 + 0/x_2 + 0/x_3 + 1/x_4 + 1/x_5$$

$$A_1 = 0/x_1 + 0/x_2 + 0/x_3 + 0/x_4 + 1/x_5$$

So applying the formula to each of the $\alpha - cuts$

$$\mu_A = \sup_{\alpha \in [0,1]} \alpha \cdot \mu_{A_\alpha}(x)$$

Gives

$$A_{0.2} = 0.2/x_1 + 0.2/x_2 + 0.2/x_3 + 0.2/x_4 + 0.2/x_5$$

$$A_{0.4} = 0/x_1 + 0.4/x_2 + 0.4/x_3 + 0.4/x_4 + 0.4/x_5$$

$$A_{0.6} = 0/x_1 + 0/x_2 + 0.6/x_3 + 0.6/x_4 + 0.6/x_5$$

$$A_{0.8} = 0/x_1 + 0/x_2 + 0/x_3 + 0.8/x_4 + 0.8/x_5$$

$$A_1 = 0/x_1 + 0/x_2 + 0/x_3 + 0/x_4 + 1/x_5$$

And when these fuzzy sets are unioned

$$A = A_{0.2} \, U \, A_{0.4} \, U \, A_{0.6} \, U \, A_{0.8} \, U \, A_1$$

The following is obtained, which is the same as original set A

$$A = \{(0.2, x_1), (0.4, x_2), (0.6, x_3), (0.8, x_4), (1, x_5)\}$$

Question 2 – Decision making in a fuzzy environment

Fuzziness can be introduced at several points in the existing models of decision making.

Question 2a

Bellman and Zadeh in 1970 suggested a fuzzy model of decisions that must accommodate certain constraints C and goals G. Provide a description of this model. **(15 marks)**

Answer 2a

In 1970 Bellman and Zadeh proposed the first fuzzy decision model in a paper titled "*Decision-making in a fuzzy environment*" [1]. Their model looked to solve the problem of making the best decision where multiple constraints (C) and goals (G) are satisfied. For example, one needs to decide on the best car from a set of alternative cars, looking to maximise G including "*sufficiently comfortable*" and "*fuel economy in the region of x*" but subject to C such as "*Not costing much more than x*" where x is a constant representing kilometres per litre and budget respectively. It is therefore clear that such imprecise G and C cannot be measured with crisp boundaries and thus are modelled with fuzzy sets. As normal, membership functions define membership degree, between 0 and 1, to which an alternative verifies C and G. Determining the membership grade of an alternative takes the intersection (logical and) of all G and C, symbolised as follows [1]:

$$\mu_D = \mu_{G_1} \cap \mu_{G_2} \cap \ldots \cap \mu_{G_n} \cap \mu_{C_1} \cap \mu_{C_2} \cap \ldots \cap \mu_{C_m}$$

The t-norm min operator is applied to ensure that the alternative satisfies all combined G and C simultaneously. However, it is reasonable to suggest that in real life some decisions can be considered valid with most, and not all, of the goals and criteria satisfied, and taking the minimum may not be suitable, therefore using averaging operators may be more relevant in certain scenarios. Note that in Zadeh's paper on decision-making, he declares that "*the most important feature of this framework is its symmetry with respect to goals and constraints – a symmetry which erases the difference between them and makes it possible to relate in a relatively simple way the concept of a decision to those of the goals and constraints of a decision process.*" [1]. In other words, with this symmetric approach, as there exists no hierarchy or ordering of the C and G themselves. They therefore play the same role and are treated equally in the determination of the decision membership grade to ensure it satisfies them both. The resulting decision is a fuzzy set of alternatives from which the best alternative can be selected by the highest minimum membership grade (max-min). Of course, the degree decision $\boldsymbol{\mu_D}$ can be used to define a linear ordering of decisions.

Question 2b

Suppose we must choose one of four different jobs a, b, c, and d, the salaries of which are given by the function f such that:

$$f(a) = 30{,}000, f(b) = 25{,}000, f(c) = 20{,}000 \text{ and } f(d) = 15{,}000.$$

Our goal is to choose the job that will give us a high salary given the constraints that the job is interesting and within close driving distance. The first constraint of interest value is represented by the fuzzy set

$$C_1 = \{(0.4, a), (0.6, b), (0.8, c), (0.6, d)\}.$$

The second constraint concerning the driving distance to each job is defined by the fuzzy set

$$C_2 = \{(0.1, a), (0.9, b), (0.7, c), (1, d)\}.$$

The fuzzy goal G of a high salary is defined by the membership function

$$\mu_G(x) = \begin{cases} 0 & for\ x < 13{,}000 \\ -0.00125\left(\frac{x}{1000} - 40\right)^2 + 1 & for\ 13{,}000 \le x \le 40{,}000 \\ 1 & for\ x > 40{,}000 \end{cases}$$

Which is the best job when applying Bellman and Zadeh's fuzzy decision model? **(10 marks)**

Answer 2b

See table I below. The salaries for each of the 4 jobs are listed as x on the second row. Below x, the constraints C_1 and C_2 with their membership grades for each of the jobs are given. The membership grade of x in G is calculated using the membership function formula provided, shown rounded to 2 decimal places.

TABLE I
JOB SALARY WITH CONSTRAINT, GOAL AND DECISION MEMBERSHIP GRADES

	a	***b***	***c***	***d***
x	30000	25000	20000	15000
C_1	0.4	0.6	0.8	0.6
C_2	0.1	0.9	0.7	1
G	0.88	0.72	0.5	0.22
μ_D	0.1	0.6	0.5	0.21

The decision membership grade can be symbolised as follows

$$\mu_D = C_1 \cap C_2 \cap G$$

Resulting in the decision as the fuzzy set below (using min operator for minimum grade that satisfies C_1, C_2 and G).

$$D = \{(30000, 0.1), (25000, 0.6), (20000, 0.5), (15000, 0.21)\}$$

Consequently, the best job according to C_1, C_2 and G is b, with salary of 25000 (max $\mu_D 0.6$).

Question 3 – ANFIS

Consider the IRIS data seen in the practical exercises. Explain how data such as this can be used to generate a fuzzy system using ANFIS. Illustrate your answer with examples from the lab work you did using ANFIS with the IRIS data. Include in your answer a brief discussion of when it is appropriate to use ANFIS to build a fuzzy system and identify some of the decisions that need to be made in order to generate a useful system. **Worth 25%**

Answer 3

Building a Mamdani-type fuzzy system typically involves configuration of membership functions (MF) where their shape (gaussian, triangular, trapezoidal etc.), properties (length, height, centre etc.) and overlap defines the fuzziness of the set. Furthermore, a knowledge-base of fuzzy IF-THEN rules are required to generate output decisions based on input. Ideally there is an expert on-hand that can leverage their problem domain insight to the specification of MF and rules criteria, but this is not always possible. Without a good understanding of the dataset the MF and rules are quite often determined heuristically and can result in an under-performing system.

The ANFIS (Adaptive Neuro-Fuzzy Inferencing System, or Adaptive-Network-Based Fuzzy Inferencing System) approach, first introduced in 1993 by Jang [2], replaces the need of a human expert and can be used to tackle both regression and classification problems. It does this by using a hybrid, deep learning technique, combining fuzzy logic with a back-propagation multi-layer perceptron (MLP) neural network (NN) to model MF and rules by multi-cycle (epoch) empirical training on a labelled dataset.

The adaptive element comes from the system's ability during training to measure the loss error between predicted and ground-truth, adjusting the weight coefficient for each input and propagating that back up the network ready for the next epoch. The MF parameters of the network are also adjusted, changing the shape of the function (Fig. 24, Appendix B). The goal is to continuously reduce the error, working towards convergence of predicted and ground-truth. Consequently, this iterative weight adjustment along with the non-linearity of membership functions (like NN activation functions) lends itself very well to resolving non-linear problems. ANFIS uses the Sugeno fuzzy model rather than Mamdani and hence generates a final layer output that "*can be expressed as a linear combinations of the consequent parameters*" [2]. Once trained ANFIS can make fuzzy decisions based on unseen input.

The well-known Iris flower dataset was compiled by British statistician and biologist Ronald Fisher in 1936 [3]. It consists of 150 samples split equally between 3 types of Iris. The length and width of the sepals and petals, measured in cm, form the 4 features. A 5th target column labels the Iris type to which the sample belongs (setosa, versicolor, virginica). This dataset was used to gain practical lab hands-on with ANFIS using the Matlab fuzzy logic toolbox [4] to resolve an Iris classification problem, and what follows is an overview of that lab work, including steps done, configuration decisions and experimentation results.

Initial dataset analysis was performed (Table II) to understand the nature of the dataset to ensure suitability for ANFIS and perhaps guide membership function selection. Given the samples are verified split equally between Iris types there is no issue here with imbalanced class distribution and risk of misclassification due to under or over representation. There is no sparse feature issue, as all rows have non-zero values. Further, there is no issue with constant value features given the number of distinct values for any feature is 22 or greater. Overall the dataset quality is good for use in ANFIS without requiring further engineering.

TABLE II
IRIS DATASET FEATURE STATISTICS

Statistic	Sepal Length	Sepal Width	Petal Length	Petal Width
# zero values	0	0	0	0
# distinct values	35	23	43	22
Min	4.3	2	1	0.1
Mean	5.843	3.054	3.759	1.199
Max	7.9	4.4	6.9	2.5
StdDev	0.828	0.434	1.764	0.763
Skew	0.315	0.334	-0.274	-0.105
Kurtosis	-0.552	0.291	-1.402	-1.340

Fig.1 presents the plotted feature distribution. *SepalLengthCm (SL)* with small skew and small negative kurtosis shows a fairly symmetrical but plateaued, platykurtic distribution where the peak is low and broad with less extreme outliers than for a normal distribution, quite trapezoidal in shape. *SepalWidthCm* (SW) with very small kurtosis and skew indicates an almost normal distribution, however tall and narrow given its small standard deviation. Both *PetalLengthCm* (PL) and *PetalWidthCm* (PW) exhibit bi-modal distribution with 2 local maximums suggesting 2 different groups at play here. This is further supported by many pair plots (Fig. 2) where 2 distinct clusters are present, in addition to linear correlation between petal length and petal width/sepal length. Note data analysis was performed using Python (Appendix A).

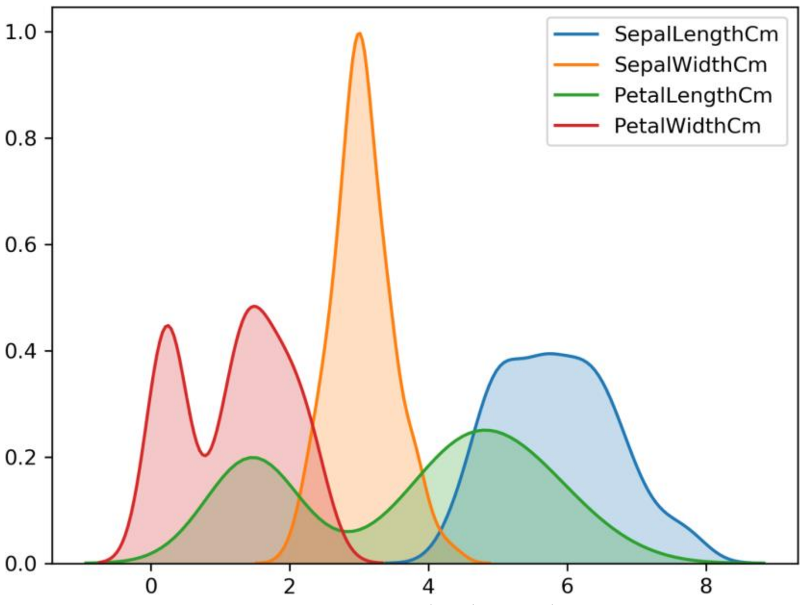


Fig. 1 Feature distribution plot

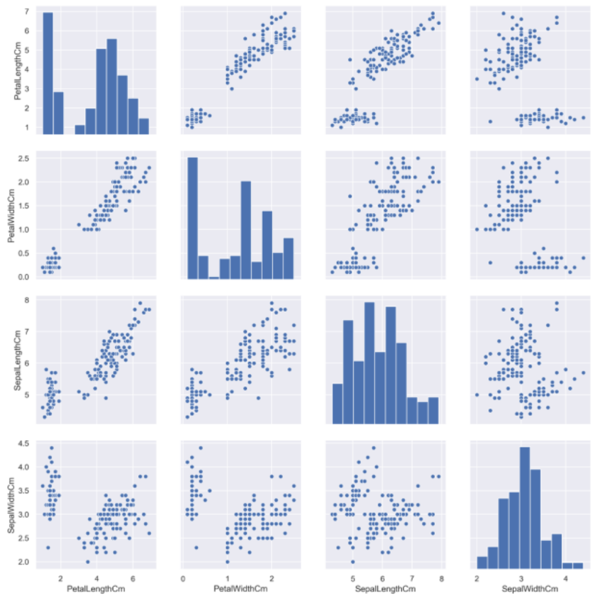


Fig. 2 Feature pair plot

The Iris dataset is partitioned into train and test with 125 and 25 samples respectively, and now it is better understood, a first Matlab ANFIS system configuration using the TRIMF triangular MF and trained with 100 epochs (Table III) establishes a prediction performance baseline. The *genfis* command [5] is used to generate a Sugeno-type FIS structure (Appendix C, ANFIS Matlab script). The high-level ANFIS topology is presented in Fig. 3, where 4 inputs are fuzzified by MF then passed to a knowledge base of rules numbering #inputs multiplied by #MF. The rule output is linearly combined in the final layer. The TRIMF MF shape for input 1 (*SepalLengthCm*) is plotted, noting the min and max values for this feature (Table II) defines the shape parameters (Fig. 4)

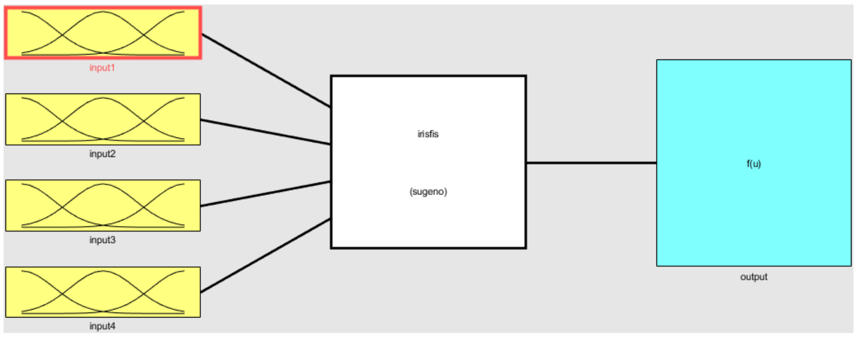
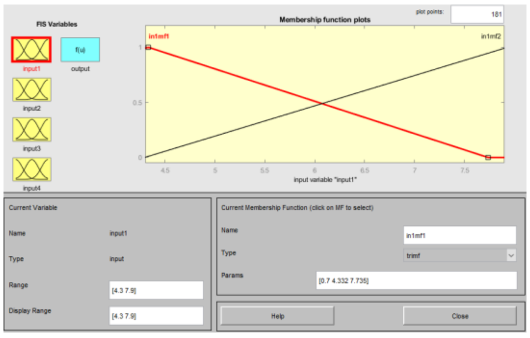


Fig. 3 ANFIS system with 4 inputs/1 output



Current Variable
Name input1
Type input
Range [4.3 7.9]
Display Range [4.3 7.9]

Current Membership Function (click on MF to select)
Name in1mf1
Type trimf
Params [0.7 4.332 7.735]

Fig. 4 Input 1 SepalLengthCm TRIMF MF

The training partition actual target and ANFIS predicted target is plotted for test 1 (Fig. 5), where the fuzzified predictions are spread around the integer actuals, with a wider distribution for classes 2 and 3 (versicolor, virginica) than for 1 (setosa). However, it is more important that a classification model can generalise on unseen data, therefore configuration performance will be judged on the RMSE (root-mean-squared error) metric comparing predicted Iris classification with actual from the test partition. Fig. 6 shows the minimum test error occurring at epoch 40, after which an increase suggests the beginning of overfitting (memorizing) data during training. With the epochs now limited in test 5 the results are marginally poorer in the unseen test data. It is the author's opinion that better results would be achieved with a larger test partition, as limiting the epochs here would help generalisation.

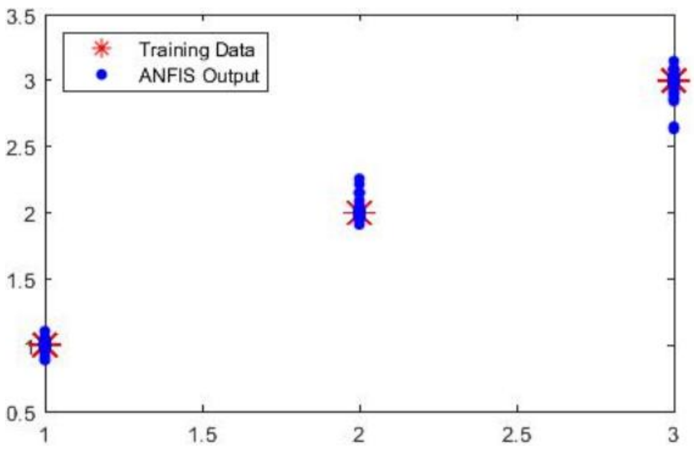


Fig. 5 Training partition actual target vs ANFIS predicted target (test 1)

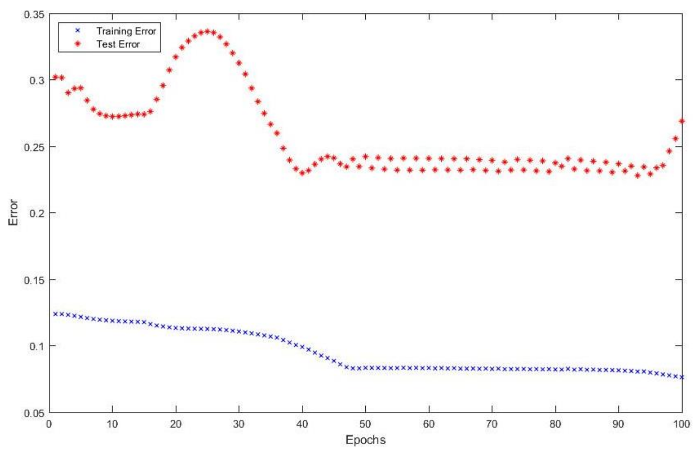


Fig. 6 Training vs test error over epochs (test 1)

Several additional configurations using all 4 features but with a different MF were tested (2, 3, 4, Table III) but were not able to yield lower RMSE than test 1. Having observed quite distinct distributions and multiple local maximums in Fig. 1 during initial dataset analysis the ANFIS configuration (test 6) was enhanced to assign to each input feature an MF whose shape reflected the feature distribution; TRAPMF (trapezoidal) for SL, GAUSSMF (gaussian) for SW, and PSIGMF (product of 2 sigmoidal) for both PL and PW (bi-modal distribution). This increased the knowledge base to 256 rules (Fig. 7) but significantly improved accuracy with a best overall RMSE of 0.1940 (down from previous best 0.2280). Test 7 produced the same test RSME as test 6 with fewer epochs with an exceptionally low training RSME (Fig. 8, Table III).

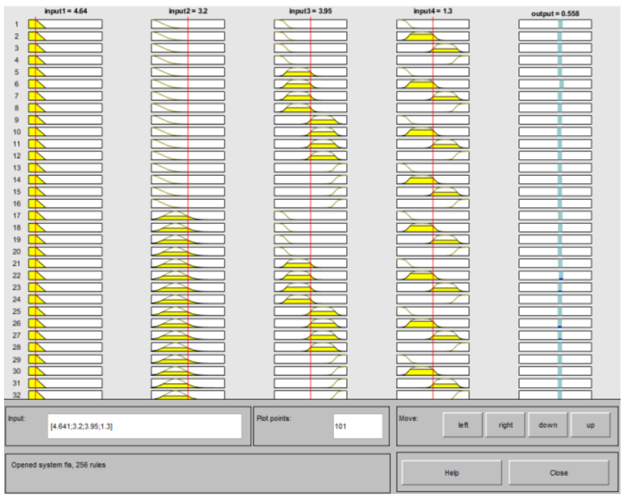


Fig. 7 Knowledge base 256 rules, specific feature MF (test 6)

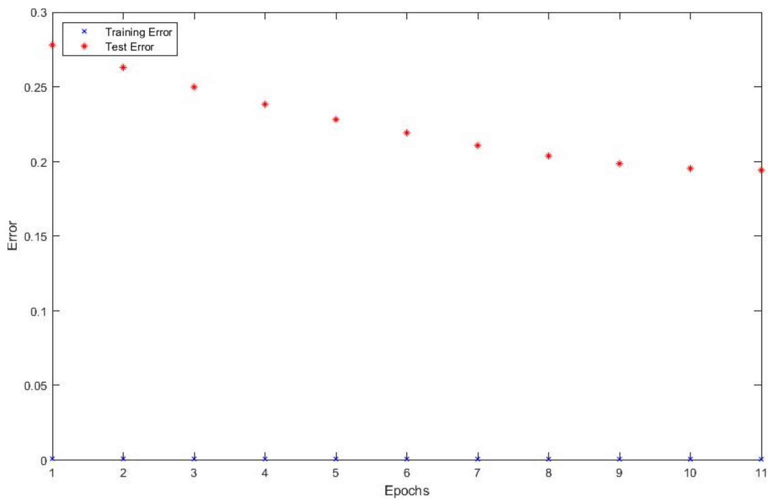


Fig. 8 Training vs test error over epochs (test 6)

As previously highlighted, features PL and PW exhibited distinctive bi-modal distributions, warranting one final ANFIS configuration, test 8, where the input space was reduced to these two features only. Despite the worst training RSME, the more important test RSME was 2nd best, with training error parallel to test error (Fig. 22, Appendix B) which indicates stability and an ability to generalise, although with 31 epochs it took longer to converge than test. Only 4 rules were required (Fig. 23, Appendix B). Summarising, if outright optimal performance was required test 7 configuration would be proposed, otherwise test 8 if simplicity that comes with minimal search and knowledge base judged good enough. Manual configuration proved straight-forward with specification of MF and epochs, ANFIS taking care of the complex MF shaping.

TABLE III
ANFIS TEST RESULTS WITH IRIS DATASET

Test #	Features	MF	# Epochs	Min Training RSME	Min Test RSME
1	All	TRIMF	100	0.07673	0.2280
2	All	GAUSSMF	100	0.0292	0.2617
3	All	TRAPMF	100	0.0137	0.5385
4	All	GBELLMF	100	0.0222	0.2770
5	All	TRIMF	40	0.0990	0.2298
6	All	TRAPMF (SL), GAUSSMF (SW), PSIGMF (PL), PSIGMF (PW)	40	0.000028209	0.1940
7	All	TRAPMF (SL), GAUSSMF (SW), PSIGMF (PL), PSIGMF (PW)	11	0.00020675	0.1940
8	PW, PL	PSIGMF (PL), PSIGMF (PW)	31	0.1343	0.2065

Question 4 – Type 2 fuzzy logic

Write a short essay (about 2 pages not including bibliography & references) to discuss why there is a perceived need for the type-2 fuzzy logic and consider the areas of application where type-2 fuzzy sets might provide a solution. Your answer should include a brief definition of a type-2 fuzzy set (illustrate with diagram(s) and example(s)). It should also make reference to issues associated with the practical implementation of type-2 fuzzy logic and to the approaches that have been developed. **Worth 25%**

Answer 4

In answering "*What computing with words means to me*" Zadeh began his response with a beautifully concise description of the human ability to "*converse, communicate, reason and make rational decisions in an environment of imprecision,* uncertainty, incompleteness of information and partiality of truth" [6] and distils his answer down to "*a methodology for reasoning, computing and decision-making with information described in natural language*" [6]. As a relative newcomer to type-1 fuzzy logic (FL), I've understood to this point that it represents the notion of uncertainty when mathematically modelling words. On researching type-2 FL for this work it came as a surprise to learn from renowned academic in the field, John Mendel , that "*to use a type-1 fuzzy set (FS) to model a word is scientifically incorrect, because a word is uncertain whereas a type-1 FS is certain*" [7]. However, with the following definition of a type-1 fuzzy set (FS) being $A = \{(x, \mu_A(x)) | x \in X\}$ where $\mu_A(x)$ has the interval [0, 1], then the membership grade output from the membership function (MF) is not fuzzy at all but a precise real number [7] and so Mendel's claim now makes sense.

Type-1 has been successfully applied in the fields of engineering (mechanical, electrical), aerospace, agricultural and robotics to name just a few of the domains, controlling air conditioners, washing machines and anti-lock braking systems to manging highway traffic, projecting risk assessments, analysing stock trades and forecasting weather [8]. These use-cases consume application-specific data where sharp membership function specification is possible due to the certainty in both the data and process, delivering satisfactory results. Yet Zadeh himself recognised issues with type-1 FS modelling domains or processes where uncertainty existed in the form of measurement error, dynamic randomness and estimate imprecision [9], including where data that helps define an MF is collected from people rather than systems which then of course inherits a natural uncertainty, where a person can be unsure of the meaning of a word (intra-uncertainty) or the differing interpretation of a word amongst a group of people (inter-uncertainty) [10].

In 1975 Zadeh extended type-1 to a type-2 FS to better model imprecision [11], which "*takes us one more step toward the goal of 'Computing with Words' or the use of computers to represent human perception*" [12]. Whereas the membership function of type-1 FS has an interval ranging [0, 1], type-2 FS add an additional 3rd dimension to the membership function with degree of membership represented by a type-1 FS, thereby offering "*an opportunity to model levels of uncertainty with which traditional fuzzy logic (type-1) struggles*" [12]. This provides a fuzzier inference system with MF shapes no longer visualised as clean lines but blurred by a variable amount, as depicted in Fig. 9. The type-1 MF is shown to return the single membership grade at the point where x slices through it, e.g. 0.8/5. Mendel and Liang introduced the characterisation of the type-2 MF with upper (UMF) and lower (LMF) bound functions [9] each of which is a type-1 MF. The area between these 2 functions is called the footprint of uncertainty (FOU) which describes the blurriness of the type-2 fuzzy set. The upper and lower MF subset the FOU max and min membership grade respectively. Note that where x slices through the FOU a fuzzy set with multiple grades is returned, e.g. 0.6/5 + 0.7/5 + 0.8/5. With this additional information "*type-2 FSs has the potential to outperform using type-1 FSs, especially when we are in uncertain environments*" [7].

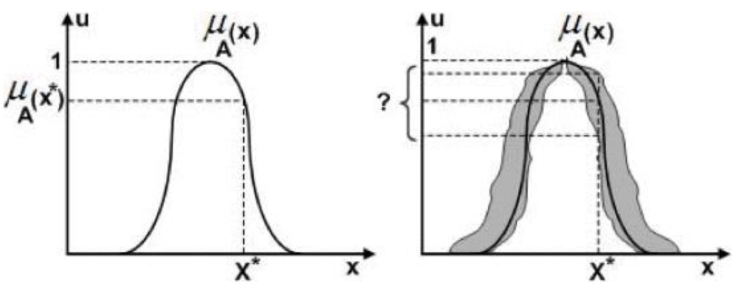


Fig. 9 Typical fuzzy membership functions: (a) type-1, (b) type-2 [13]

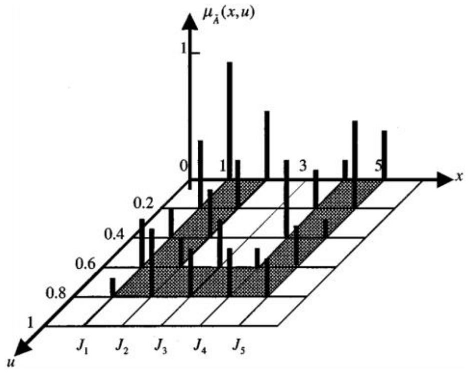


Fig. 10 Example general type-2 MF. Shaded area is the FOU [14]

"*General*", "*embedded*" and "*interval*" type-2 fuzzy sets are now briefly discussed. The general type-2 set is formalised as follows, with the tilde ~ denoting A is a type-2 fuzzy set:

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) | \forall x \in X, \forall u \in J_x \subseteq [0,1]\}$$

Consider Fig. 10 illustrating a general type-discretized fuzzy set, with primary membership each x, u intersection with the secondary membership grades (3rd dimension) represented by varying stick heights in closed interval [0, 1] sitting atop the FOU at every x, u intersection. The J_x lines are the primary memberships of X, for example J_1 = [0,0.2,0.4,0.6,0.8]. With secondary membership defined as $\mu_{\tilde{A}}(x,u)$, then $\mu_{\tilde{A}}(1, 0.8)$ = 0.1 and $\mu_{\tilde{A}}(4, 0.4)$ = 0.3. The full set comprises the union of all possible primary membership sets with their secondary grades that clearly adds a rich fuzzy expressiveness to the uncertainty, but computationally expensive. This is due to the high cardinality, with the possible sets the sum of number of secondary memberships on each J_x multiplied by each other i.e. 5 x 5 x 2 x 5 x 5 = 1250 type-2 sets for Fig. 10. Such complexity and cost limits adoption, with the authors of [15] stating, "*So far, generalised type-2 fuzzy applications are few in number. This is attributable to the enormous computational complexity of generalised type-2 fuzzy inferencing*".

Several different approaches have looked to reduce complexity and computing cost of type-2 systems, including a model known as embedded type-2 sets. Fig. 11 shows 2 embedded (or wavy slice) atop a shaded FOU. Sets Ã and Ĩ are indicated by the green and blue flags respectively. As per general sets, the secondary grades are given by the flag height. Whereas for general sets each x, u intersection contained a stick, only a single secondary domain value u is present on each primary membership J_x line. Therefore, for embedded set Ã we have

$$\tilde{A} = \{[0.1/0]/0 + [0.1/0.1]/0.1 + [0.5/0.4]/0.2 + [0.5/0.1]/0.3 + [1/1]/0.4 + [1/0.6]/0.5 + [0.4/0]/0.6 + [0.5/0.2]/0.7 + [0.1/0.2]/0.8 + [0.1/0]/0.9\}$$

The significant difference between general and embedded type-2 fuzzy systems, and a distinct advantage in the latter's favour, is the reduced set cardinality, especially important in output processing, as only a subset or samples of the full type-2 are used. Note that the union of all possible embedded sets forms the generalised type-2 fuzzy set as proved by the Fundamental Decomposition Theorem [14].

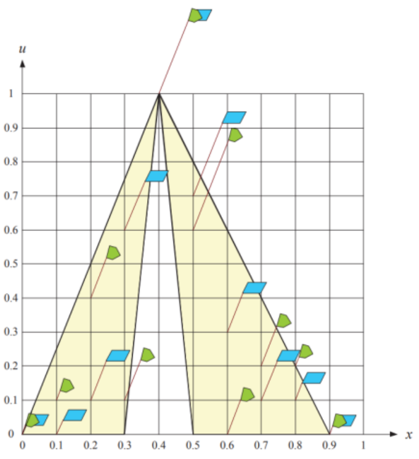


Fig. 11 2 embedded type-2 sets, FOU shaded [16]

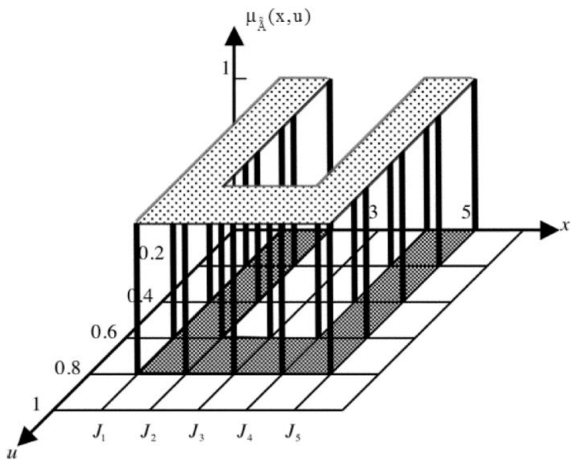


Fig. 12 Example interval type-2 MF [17]

Introduced in 2000 by Liang and Mendel [18], interval fuzzy sets are another approach to simplifying type-2 systems. Fig. 12 illustrates the transformation of each secondary membership grade stick height shown in Fig. 10 to a value of 1, with $\mu_{\tilde{A}}(1, 0.8)$ = 1 as an example. As the heights for secondary membership grades are all 1 for each J_x we can simplify by specifying a crisp interval range rather than discrete primary memberships. Therefore, J_1 = [0,0.8] and given a constant height of 1 can be considered a plane starting at 0 and ending at 0.8. With such intervals the computational expense of output operations is reduced. Mendel states that interval sets "*are the most widely used type-2 fuzzy sets to date*" [14] and in fact suggests them as a simplification approach when trying to determine optimal appropriate MF for type-2 systems by proposing to "*Begin by specifying the FOU. Then, instead of using arbitrary possibilities at each point of the FOU, use the same possibility over the entire FOU*" [14] as "*is in general no single best choice for a type-1 MF, it seems a bit foolhardy (to me) to believe that at each value of the primary variable, x, there is some optimal secondary MF*" [14].

In the context of fuzzy type-2 performance and implementation it is relevant to take a brief look at the topology of a Fuzzy Inferencing System (FIS, Fig. 13), in particular the output step which consists of 2 components. The responsibility of the type-reducer, an additional component not present in type-1 systems, is to transform type-2 fuzzy sets to type-1 fuzzy sets. A number of different type reduction methods are available, including "*centroid, center-of-sets, height, and modified height*" [18]. At this point the result may be output as the "*type reduced set may be more important than a single crisp number since it conveys a measure of uncertainties that have flown through the type-2 FLS*" [19] before a further transformation by defuzzification to crisp type-0 number values that can be more readily used in applications. It is desirable to leverage typical, general type-2 systems, given their ability to deal better with uncertainty, as their secondary membership grades are fuzzy. However, they are computationally expensive and slow and thus a challenge to implement, as type reduction needs to process all embedded type-2 sets [20] [18] [14], having to find the centroid or modified height in a large number of type-1 sets which bottlenecks the entire system [20]. Even the simple example illustrated in Fig. 10 requires processing 1250 sets. As Greenfield et al state, this computing constraint "*has hampered the development of type-2 Fuzzy Inferencing Systems for real applications and therefore no advantage has been taken of the ability of type-2 fuzzy sets to model higher levels of uncertainty*" [20].

Research looking to resolve the intensity of type-reduction includes a novel approach that randomly samples and processes a subset of embedded sets from the full type-2 set, with the authors reporting greatly reduced processing time at cost of only a little loss in accuracy [20]. For example, exhaustively processing all embedded sets in the "*shopping FIS*" test took 17.64s returning a defuzzified value of 0.595, while the 100 sample variation took only 0.09s with defuzzified value of 0.593. With regard to the defuzzification of interval type-2 sets, the same authors in [19] report improvements for accuracy (asymmetric test sets) and speed (both symmetric/asymmetric) using RESA (Representative Embedded Set Approximation) methods over the typically used Karnik-Mendel iterative procedure (KMIP).

Despite challenges in implementation, type-2 fuzzy controllers offer improved performance for applications that require better uncertainty handling [21] and there are many documented examples. One such use case is the control of large, powerful and expensive marine diesel engines where type-2 controllers minimise error deviation from setpoint despite influencing sources of uncertainty including sensor noise, vibration, frequency interference, mechanical wear and load that varies according to weather and sea conditions [21]. In the energy sector a type-2 fuzzy system was used to forecast short-term electricity demand with acceptable accuracy, outperforming tested neural network models [22]. In the domain of cyber security, the enriched uncertainty within type-2 systems proved better able to detect computer network anomalies [22]. Type-2 controllers were also found to perform better than their type-1 counterparts in outdoor robots navigating unstructured environments in varying weather, surface and lighting conditions [21]. This was also the case in [23] where the trajectory of a robot following a path guided by different type-n controllers was plotted (Fig. 14). The type-1 controller shows greatest error variance due to lack of fuzziness in dealing with an uncertain environment. The type-2 variants do better, with generalised showing the best accuracy, suggesting a greater finesse in output control when navigating uncertainty thanks to the richer internal fuzziness of this model.

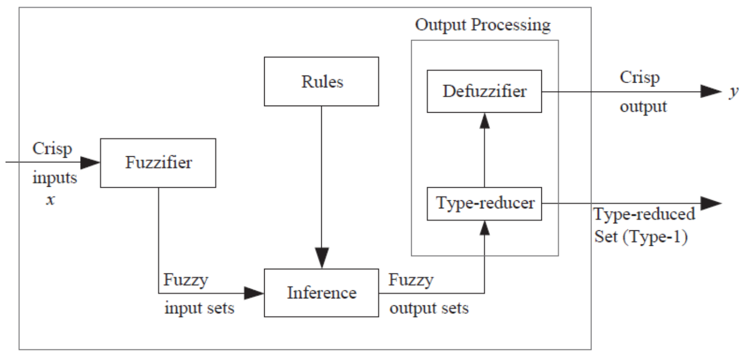


Fig. 13 Type-2 FIS [19]

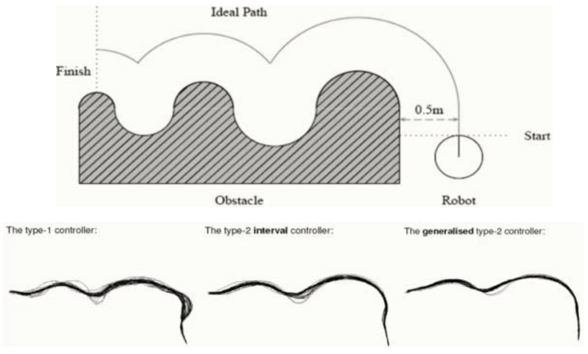

Ideal Path
Finish
0.5m
Start
Obstacle
Robot
The type-1 controller:
The type-2 interval controller:
The generalised type-2 controller:

Fig. 14 Robot navigation path, different fuzzy-n controllers [23]

References

[1] R. Bellman and L. Zadeh, “Decision-Making in a Fuzzy Environment,” *Management Science,* pp. B-141 - B-164, 1970.

[2] J. Jang, “ANFIS: Adaptive-Network-Based Fuzzy Inference System,” *IEEE transactions on systems, man and cybernetics,* vol. 23, no. 3, 1993.

[3] R. Fisher, “The use of multiple measurements in taxonomic problems,” 1936.

[4] MathWorks, “Fuzzy Logic Toolbox - MATLAB,” [Online]. Available: https://ch.mathworks.com/products/fuzzy-logic.html.

[5] MathWorks, “Generate Fuzzy Inference System Structure from data,” [Online]. Available: https://ch.mathworks.com/help/fuzzy/genfis.html.

[6] J. M. Mendel, L. A. Zadeh, E. Trillas, R. Yager, J. Lawry, H. Hagras and S. Guadarrama, “What Computing with Words Means to Me,” *IEEE computational intelligence magazine,* pp. 20-26, 2010.

[7] J. M. Mendel, “Type-2 Fuzzy Sets:Some Questions and Answers,” *IEEE Neural Networks Society,* pp. 10-13, 2003.

[8] H. Singh, M. Gupta, T. Meitzler, Z.-G. Hou, K. K. Garg, A. Solo and L. Zadeh, “Real-Life Applications of Fuzzy Logic,” *Advances in fuzzy systems,* vol. 2013, 2013.

[9] O. Castillo, P. Melin, J. Kacprzyk and W. Pedrycz, “Type-2 Fuzzy Logic: Theory and Applications,” in *2007 IEEE International Conference on Granular Computing (GRC 2007)*, Fremont, CA, USA, 2007.

[10] J. M. Mendel, “Fuzzy Sets for Words: a New Beginning,” in *The 12th IEEE International Conference on Fuzzy Systems, 2003. FUZZ '03*, St Louis, MO, USA, USA, 2003.

[11] L. Zadeh, “The Concept of a Linguistic Variable and its Application to Approximate Reasoning,” *Information Sciences,* vol. 8, pp. 199-249, 1975.

[12] R. John and S. Coupland, “Type-2 Fuzzy Logic: A Historical View,” *IEEE Computational Intelligence Magazine,* vol. 2, no. 1, pp. 57-62, 2007.

[13] S. Mohagheghi, G. Venayagamoorthy and R. Harley, “An Interval Type-II Robust Fuzzy Logic Controller for a Static Compensator in a Multimachine Power System,” in *The 2006 IEEE International Joint Conference on Neural Network Proceedings*, Vancouver, BC, Canada, 2006.

[14] J. Mendel and R. John, “A Fundamental Decomposition of Type-2 Fuzzy Sets,” in *Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference (Cat. No. 01TH8569)*, Vancouver, BC, Canada, 2001.

[15] S. Greenfield and F. Chiclana, “Slicing Strategies for the Generalised Type-2 Mamdani Fuzzy Inference System,” in *15th International Conference on Artificial Intelligence and Soft Computing*, Zakopane, Poland, 2016.

[16] S. Greenfield and R. John, “Stratification in the Type-Reduced Set and the Generalised Karnik-Mendel Iterative Procedure,” in *Intelligent Processing and the Management of Uncertainty (IPMU)*, Malaga, Spain, 2008.

[17] S. Senturk, N. Erginel and Y. Binici, “Interval Type-2 Fuzzy Analytic Network Process for Modelling a Third-Party Logistics (3PL) Company,” *J. of Mult.-Valued Logic & Soft Computing,* vol. 28, pp. 311-333, 2017.

[18] Q. Liang and J. M. Mendel, "Interval Type-2 Fuzzy Logic Systems: Theory and Design," *IEEE transactions on fuzzy systems,* vol. 8, no. 5, pp. 535-550, 2000.

[19] S. Greenfield, F. Chiclana, S. Coupland and R. John, "The collapsing method of defuzzification for discretised interval type-2 fuzzy sets," *Information Sciences,* vol. 179, pp. 2055-2069, 2009.

[20] S. Greenfield, F. Chiclana, R. John and S. Coupland, "The Sampling Method of Defuzzification for Type-2 Fuzzy Sets: Experimental Evaluation," *Information Sciences,* vol. 189, pp. 77-92, 2012.

[21] H. Hagras and C. Wagner, "Introduction to Interval Type-2 Fuzzy Logic Controllers - Towards Better Uncertainty Handling In Real World Applications," *IEEE Systems, Man and Cybernetics Society,* no. 27, 2009.

[22] H. Hagras and C. Wagnmer, "Towards the wide spread use of type-2 fuzzy logic systems in real world applications," *IEEE Computational Intelligence Magazine,* vol. August 2012, pp. 14-24, 2012.

[23] O. Castillo, "Type-2 Fuzzy Logic in Intelligent Control," [Online]. Available: https://www.youtube.com/watch?v=nzaWbermvDc&t=972s.

APPENDIX A – IRIS_ANALYSIS.PY

```
import time
from contextlib import contextmanager
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

@contextmanager
def timer(title):
    t0 = time.time()
    yield
    print('{} - done in {:.0f}s'.format(title, time.time() - t0))


class IrisFis:
    def __init__(self):
        self.random_state = 20
        self.column_stats = {}
        self.iris_test = None
        self.iris_train = None
        self.iris_full = None

        self.label_map_string_2_int = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}

        with timer('\nPreparing dataset'):
            self.load_data()

        with timer('\nStatistics'):
            self.column_statistics()
            self.row_count_by_target('Species')
            self.distribution()
            self.pair_plot()

    def load_data(self):
        self.iris_test = pd.read_csv('data/iristest.dat', header=None)
        self.iris_test.columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species']
        self.iris_train = pd.read_csv('data/iristrain.dat', header=None)
        self.iris_train.columns = ['SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm', 'Species']
        self.iris_full = pd.concat([self.iris_test, self.iris_train], ignore_index=True)

        print('\tRow count:\t', '{}'.format(self.iris_full.shape[0]))
        print('\tColumn count:\t', '{}'.format(self.iris_full.shape[1]))

    def column_statistics(self):
        print('\n--- Column Stats')
        for col in self.iris_full:
            self.column_stats[col + '_dtype'] = self.iris_full[col].dtype
            self.column_stats[col + '_zero_num'] = (self.iris_full[col] == 0).sum()
            self.column_stats[col + '_zero_pct'] = (((self.iris_full[col] == 0).sum() / self.iris_full.shape[0]) * 100)
            self.column_stats[col + '_nunique'] = self.iris_full[col].nunique()
            print('\n- {} ({})'.format(col, self.column_stats[col + '_dtype']))
            print('\tzero {} ({:.2f}%)'.format(self.column_stats[col + '_zero_num'],
                                               self.column_stats[col + '_zero_pct']))
            print('\tdistinct {}'.format(self.column_stats[col + '_nunique']))

            # Numerical features
            if self.iris_full[col].dtype != object:
                self.column_stats[col + '_min'] = self.iris_full[col].min()
                self.column_stats[col + '_mean'] = self.iris_full[col].mean()
                self.column_stats[col + '_quantile_25'] = self.iris_full[col].quantile(.25)
                self.column_stats[col + '_quantile_50'] = self.iris_full[col].quantile(.50)
                self.column_stats[col + '_quantile_75'] = self.iris_full[col].quantile(.75)
                self.column_stats[col + '_max'] = self.iris_full[col].max()
                self.column_stats[col + '_std'] = self.iris_full[col].std()
                self.column_stats[col + '_skew'] = self.iris_full[col].skew()
                self.column_stats[col + '_kurt'] = self.iris_full[col].kurt()
                print('\tmin {}'.format(self.column_stats[col + '_min']))
                print('\tmean {:.3f}'.format(self.column_stats[col + '_mean']))
                print('\t25% {:.3f}'.format(self.column_stats[col + '_quantile_25']))
                print('\t50% {:.3f}'.format(self.column_stats[col + '_quantile_50']))
                print('\t75% {:.3f}'.format(self.column_stats[col + '_quantile_75']))
                print('\tmax {}'.format(self.column_stats[col + '_max']))
                print('\tstd {:.3f}'.format(self.column_stats[col + '_std']))
                print('\tskew {:.3f}'.format(self.column_stats[col + '_skew']))
                print('\tkurt {:.3f}'.format(self.column_stats[col + '_kurt']))

    def distribution(self):
        for col in self.iris_full:
            if col != 'Species':
                sns.kdeplot(self.iris_full[col], shade=True)

        plt.savefig(fname='plots/iris distplot.png', dpi=300, format='png')
        plt.show()

    def pair_plot(self):
        sns.set()
        cols = ['PetalLengthCm', 'PetalWidthCm', 'SepalLengthCm', 'SepalWidthCm']
        sns.pairplot(self.iris_full[cols], height=3)
        plt.savefig(fname='plots/iris pairplot.png', dpi=300, format='png')
        plt.show()

    def row_count_by_target(self, target):
        print('\n--- Row count by {}'.format(target))
        series = self.iris_full[target].value_counts()
        for idx, val in series.iteritems():
            print('\t{}: {} ({:6.3f}%)'.format(idx, val, ((val / self.iris_full.shape[0]) * 100)))


irisFis = IrisFis()
```

APPENDIX B – ANFIS - TRAINING VS TEST ERROR OVER EPOCHS TEST PLOTS, RULES, MF PARAMETERS

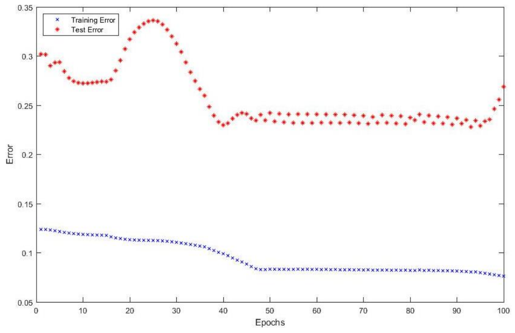


Fig. 15 Test 1 - All features, TRIMF, 100 epochs

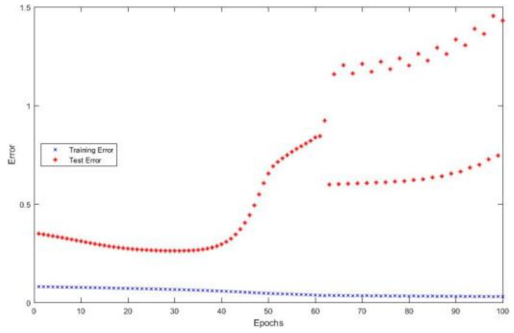


Fig. 16 Test 2 – All features, GAUSSMF, 100 epochs

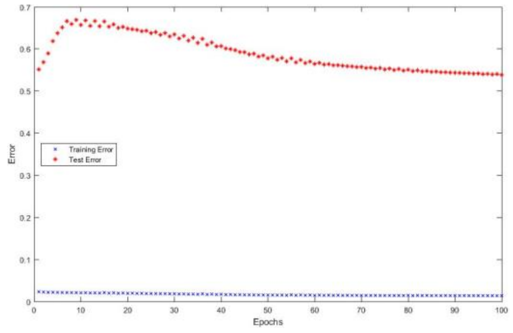


Fig. 17 Test 3 – All features, TRAPMF, 100 epochs

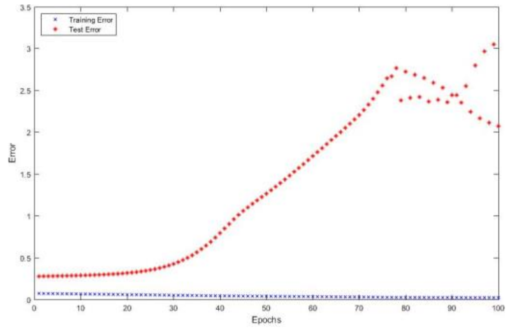


Fig. 18 Test 4 – All features, GBELLMF, 100 epochs

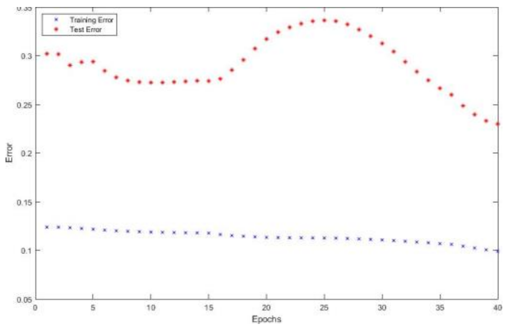


Fig. 19 Test 5 - All features, TRIMF, 40 epochs

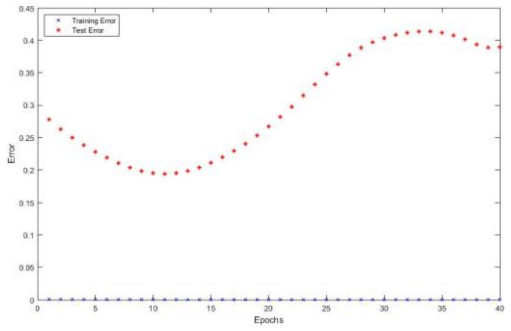


Fig. 20 Test 6 – All features, multi-mf, 40 epochs

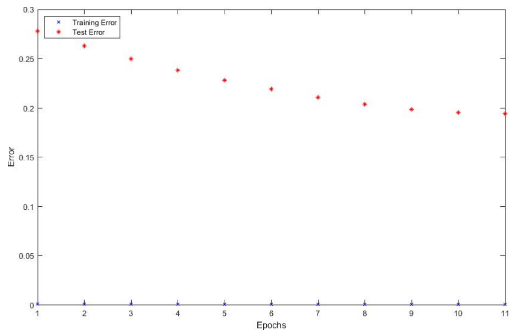


Fig. 21 Test 7 – All features, multi-mf, 11 epochs

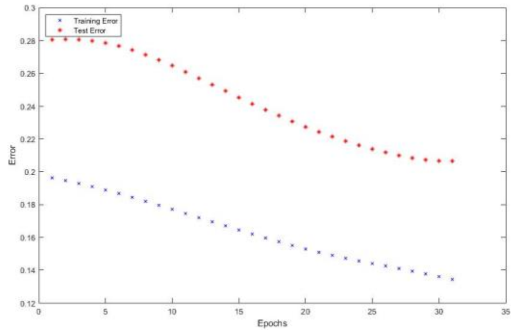


Fig. 22 Test 8 – Features PW/PL, PSIGMF, 31 epochs

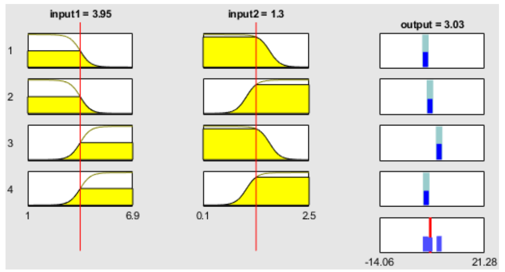
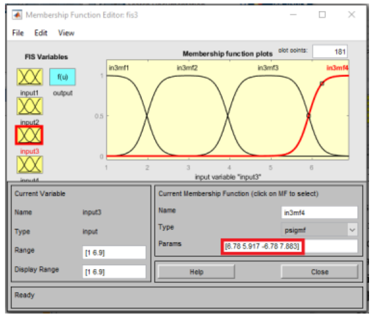


Fig. 23 Test 8 - Features PW/PL, PSIGMF, 4 rules



plot points: 181

Current Variable

Name input3

Type input

Range [1 6.9]

Display Range [1 6.9]

Current Membership Function (click on MF to select)

Name in3mf4

Type psigmf

Params [6.78 5.917 -6.78 7.883]

Ready

Fig. 24 ANFIS-adjusted MF parameters

APPENDIX C – ANFIS – MATLAB SCRIPT

```
% Working directory is data
load iristrain.dat
load iristest.dat

% Test 1
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 2;
genOpt.InputMembershipFunctionType = 'trimf';
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=100
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest

[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);
y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 10')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','NorthWest')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 2
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 2;
genOpt.InputMembershipFunctionType = 'gaussmf';
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=100
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')
```

```
x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','West')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 3
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 2;
genOpt.InputMembershipFunctionType = 'trapmf';
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=100
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','West')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 4
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 2;
genOpt.InputMembershipFunctionType = 'gbellmf';
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=100
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
```

```
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','NorthWest')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 5
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 2;
genOpt.InputMembershipFunctionType = 'trimf';
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=40
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','NorthWest')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 6
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 4;
genOpt.InputMembershipFunctionType = ["trapmf","gaussmf","psigmf","psigmf"]
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=40
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
```

```
opt.ValidationData=iristest
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','NorthWest')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 7
x=iristrain(:, 1:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 4;
genOpt.InputMembershipFunctionType = ["trapmf","gaussmf","psigmf","psigmf"]
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=11
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','NorthWest')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)


% Test 8
x=iristrain(:, 3:4)
y_actual=iristrain(:,5)
genOpt = genfisOptions('GridPartition');
genOpt.NumMembershipFunctions = 2;
genOpt.InputMembershipFunctionType=["psigmf","psigmf"]
```

```
irisFIS = genfis(x,y_actual,genOpt);

numberEpochs=31
opt = anfisOptions('InitialFIS',irisFIS,'EpochNumber', numberEpochs);
opt.ValidationData=iristest(:, 3:5)
[outFIS,trainError,stepSize,chkFIS,testError] = anfis([x y_actual],opt);

y_predict=evalfis(outFIS,x)
figure('Renderer', 'painters', 'Position', [10 10 500 300])
plot(y_actual,y_actual,'*r',y_actual,y_predict,'.b', 'MarkerSize', 15')
legend('Training Data','ANFIS Output','Location','NorthWest')

x=[1:numberEpochs];
figure('Renderer', 'painters', 'Position', [10 10 1000 600])
plot(x,trainError,'Xb',x,testError,'*r', 'MarkerSize', 5')
legend('Training Error','Test Error','Location','NorthWest')
xlabel('Epochs')
ylabel('Error')

fisTrainRMSE = min(trainError)
fisTestRMSE = min(testError)
```