How the Learning of Rule Weights Affects the Interpretability of Fuzzy Systems

Detlef Nauck and Rudolf Kruse
University of Magdeburg, Faculty of Computer Science,
Universitaetsplatz 2, D-39106 Magdeburg, Germany
Tel: +49.391.67.12700, Fax: +49.391.67.12018
E-Mail: Detlef.Nauck@cs.uni-magdeburg.de, WWW: fuzzy.cs.uni-magdeburg.de/~nauck

Abstract

Neuro-fuzzy systems have recently gained a lot of interest in research and application. These are approaches that learn fuzzy systems from data. Many of them use rule weights for this task. In this paper we discuss the influence of rule weights on the interpretability of fuzzy systems. We show how rule weights can be equivalently replaced by modifications in the membership functions of a fuzzy system. By this we elucidate the effects rule weights have on a fuzzy rule base. Using our neuro-fuzzy model NEFCLASS we demonstrate at a simple example the problems of using rule weights, and we show, that learning in fuzzy systems can be done without them.

1. Introduction

Learning techniques are widely used to support the development of fuzzy systems. Approaches that learn fuzzy rules and/or membership functions from data by supervised or reinforcement learning are usually called neuro-fuzzy systems [9].

The most simple way to tune a fuzzy system is to use rule weights. This approach is very often used in commercial fuzzy software, and also in some neuro-fuzzy models [6], [9]. In this paper we show how rules weights can be replaced equivalently by modifying the membership functions of a fuzzy rule instead. If this is done, the effect of a weight to the semantics of a fuzzy rule becomes visible. It turns out that weights implicitly cause membership functions to change in a way that they often cannot be interpreted linguistically anymore. Thus, rule weights can completely destroy the interpretability of the fuzzy system. In this paper we mean by interpretability or readability of fuzzy systems that the membership functions are normal, convex and that they suitably cover the domain of a variable. The main idea is that a user of a fuzzy system should be able to label each fuzzy set a suitable linguistic term. Another important issue is that each linguistic term is represented by only one membership function in the fuzzy system. However, we do not want to discuss formal aspects of fuzzy partitions to ensure interpretability. What we want to show is that if there is a interpretable fuzzy system, then the interpretability is lost once rule weights are used.

Note that we consider fuzzy systems that are used for function approximation (special cases are for example fuzzy classification and fuzzy control). Such fuzzy systems consist of rules that must not be interpreted as logical rules. Each fuzzy rule is a vague sample i.e. a multidimensional fuzzy set in the data space. Applying a rule weight to such a fuzzy rule means to change the representation of this multidimensional fuzzy set. The influence of a rule weight on the interpretation of such a modified rule becomes visible in its projections which are the one-dimensional membership functions of the individual variables.

Weighted rules are also considered in probabilistic or possibilistic settings which are not within the scope of this paper. For example, certainty factors are an early heuristic approach to use rule weights for modeling beliefs. However, it turned out that the original certainty factor approach is inconsistent [5], [8]. For modeling degrees of belief or truth in fuzzy knowledge-based systems refer e.g. to [1], [2], [8], [7].

2. Interpretation of Rule Weights

A weighted rule is often written by appending "with w" to it, where w is a real value that can be different for each rule, e.g.: if x is A and y is B then z is C with w, where A, B and C are fuzzy sets of the real line, x and y are input variables, z is the output variable and w is a real-valued rule weight. The meaning of the "with-operation" that ties a weight to a rule must be defined. We will only consider the following two cases:
1. **Rule weights are applied to complete rules**: The antecedent of a rule is evaluated to determine a degree of fulfilment which is then multiplied by a rule

weight.

2. **Rule weights are applied only to the consequent parts of rules**: The degree of fulfilment of a rule is used to compute the conclusion which is then multiplied by a rule weight. If the conclusion is a fuzzy set, its support is changed this way.

Using rule weights gives rise to some semantical problems [9]. The weighting of rules is sometimes interpreted as a measure of "importance", "influence" or "reliability". However, if a rule is less "important", one usually means something like that it is only seldom applicable, or that it is not harmful if the rule is not applied, but not that its consequent should be taken into account only to some extent. This aspect is already modeled by using fuzzy sets to describe the antecedent.

To view a rule weight as "influence" could mean that a rule with a small weight should only marginally contribute to the system output. Allowing the weights to be selected from $[0, 1]$ could be interpreted as something like a degree of support for a rule. A value less than 1 would denote something like an ill-defined rule that supports its consequent only to some extent. Some approaches allow the weights to assume any value in $\mathbb{R}$, but one would obviously leave the semantics of fuzzy rules behind, because it is not clear how rules weighted by absolute values greater than 1 or by negative values should be interpreted. The influence of a rule should be modeled either by modifying its antecedent such that the rule does only apply to a certain degree in the situations considered, or by choosing an appropriate consequent that explicitly represents the rule's part in the system output.

The interpretation of a rule weight as "reliability" or "trust" is also questionable. If we belief that a rule is reliable only to a certain extent, then this means that we can only trust the final conclusion to a certain extent. However, simply multiplying the output of a rule or its degree of fulfilment by some weight is not an appropriate way to model our belief. For this probabilistic [8] or possibilistic [7] methods should be considered.

Often rule weights are used to obtain exact values in the output of a fuzzy system and therefore a weight can assume any value. In this case it is possible that negative rule weights occur that are sometimes used to represent negative rules [4]. The interpretation of a rule with a negative weight is especially difficult. It must not be interpreted as a negative proposition, but as a local sample with negative function result [9].

Let us take a closer look at this interpretation. The kind of fuzzy systems that we consider in this paper do not represent rules in a logical sense. The rules are vague samples that are used to approximate an otherwise unknown function. With this aspect in mind we have to consider the meaning of a "negative rule". If we choose to interpret it by "**if not** ... **then** ...", it would not have the character of a local sample. It would correspond to a global description of the function:

if the input is **outside** of the area
    specified by the antecedent,
then the function yields the value
    represented by the conclusion.

By overlapping with other rules a negative rule would have the effect of an offset for the function.

However, a rule with a negative weight cannot be interpreted as a proposition in the above-mentioned sense, but only as a local sample with negative function result. In a fuzzy classification system such a rule can be interpreted as a rule with negative consequent (**if** ... **then not** ...). By this a weight has an inhibitive influence on the selection of a certain class.

For a user the correct understanding of negative rules is very important. Without this knowledge it is not possible to initialize the system properly or to interpret the learning outcome.

Depending on whether a rule weight is multiplied to the degree of fulfilment or to the output of a rule, weighting a fuzzy rule is actually equivalent to changing its antecedent or consequent, respectively. Therefore rule weights can always be replaced by changes in the fuzzy sets of a rule. As we will show in the following section, these changes can lead to non-normal fuzzy sets and to the fact that identical linguistic values are represented in different ways in different rules.

3. Influence of Rule Weights

We consider only rule weights for Mamdani-type fuzzy systems in greater detail. In Sugeno-type fuzzy systems, weights which are applied to the complete rule have the same influence as in Mamdani-type fuzzy systems. If the weights are applied to the consequents there are no semantical problems, because the consequents of Sugeno-type rules are functions or constants that are not interpreted linguistically.

For the sake of simplicity we consider the following two Mamdani-type fuzzy rules for the remaining text to discuss the influence of rule weights:

$$M_1\text{: if } x \text{ is } A \text{ and } y \text{ is } B \text{ then } z \text{ is } D \text{ with } w_1,$$
$$M_2\text{: if } x \text{ is } A \text{ and } y \text{ is } C \text{ then } z \text{ is } D \text{ with } w_2,$$

where A, B, C and D are fuzzy sets of the real line, x and y are input variables, z is the output variable and w_i is a rule weight. Each fuzzy set is labeled with an individual linguistic term. Note that both rules use fuzzy sets A and D.

For Sugeno-type fuzzy systems we consider rules of the form

S_1: if x is A and y is B then $z = f_1(x, y)$ with w_1,
S_2: if x is A and y is C then $z = f_2(x, y)$ with w_2,

where f_i is a function over the input variables and w_i is a rule weight. The overall output for z is computed by a weighted sum. Note that both rules use fuzzy set A.
1. If a rule weight is used to modify the degree of fulfilment of a rule we can equivalently replace it by changing the membership functions in the antecedents. Let us consider the two most common cases, where the degree of fulfilment τ is computed either by a minimum operation or by a product.
In the case of a minimum operation we obtain for the weighted degree of fulfilment

$$\begin{aligned} w \cdot \tau &= w \cdot \min\{\mu_1(x_1), \ldots, \mu_n(x_n)\} \\ &= \min\{w \cdot \mu_1(x_1), \ldots, w \cdot \mu_n(x_n)\}. \end{aligned}$$

This means we can replace the rule weight equivalently by multiplying the individual membership degrees, i.e. we scale the heights of the fuzzy sets of the antecedent with the rule weight.
If a product is used to determine the degree of fulfilment, we can distribute the rule weights over the membership functions, e.g.:

$$w \cdot \tau = w \cdot \prod_{i=1}^{n} \mu_i(x_i) = \prod_{i=1}^{n} \sqrt[n]{w} \cdot \mu_i(x_i).$$

This again results in scaling the heights of the fuzzy sets. Different solutions are possible here, and in case of negative rule weights it is especially obvious that a rule weight can be equivalently replaced by a large number of possible fuzzy set modifications leading to a large number of different interpretations. For other t-norms similar replacements of rule weights can be found.
In any case, a rule weight leads to a modification of the membership degrees of the antecedent fuzzy sets. Therefore we only consider the case where the degree of fulfilment is determined by the minimum. In this case the effect of a rule weight can be nicely demonstrated. Using our two rules M_1 and M_2 from above we obtain the following two modified rules:

M_1^*: if x is A' and y is B' then z is D,
M_2^*: if x is A'' and y is C'' then z is D.

But now we have for the fuzzy sets in the antecedents:

$$A', B' \; : \; \begin{cases} \mathbb{R} \longrightarrow [0, w_1] & \text{if } w_1 > 0 \\ \mathbb{R} \longrightarrow [w_1, 0] & \text{otherwise} \end{cases}$$

$$A'', C'' \; : \; \begin{cases} \mathbb{R} \longrightarrow [0, w_2] & \text{if } w_2 > 0 \\ \mathbb{R} \longrightarrow [w_2, 0] & \text{otherwise} \end{cases}$$

If we assume $v \neq w \neq 1$, we obtain the following problems:

- Instead of using the same fuzzy set A the rules now use two different fuzzy sets A' and A'' in their antecedents. However, both fuzzy sets are labeled with the same linguistic term, i.e. we now have two different representations for the same linguistic term within the fuzzy system. This is of course highly undesirable if we want to interpret the rule base.
- The resulting fuzzy sets A' and A'' are not normal anymore. The application of the rule weights resulted in rescaling their membership degrees. This has also undesirable effects on the readability of the fuzzy systems. Although the interval $[0, 1]$ is arbitrarily chosen to represent membership degrees, we have to keep in mind that rule weights imply a different interval of membership degrees for each rule.

Interpretation is affected most, if the weights are larger than 1 or smaller than 0. In this case strictly spoken A' and A'' are no longer fuzzy sets. Especially, the meaning of negative membership degrees is not clear at all.
In Mamdani-type fuzzy systems rule weights that are used to change the degree of fulfilment of the rules have different effects depending on the kind of implication used. Let us consider only the two most common cases:

- If we use min-implication to compute the conclusion of a rule, then $w > 1$ only has an effect if $w\tau < 1$ holds. If $w\tau > 1$ holds, the consequent is not modified. Weights from $[0, 1]$ result in degrees of fulfilment from $[0, 1]$, and so the computation of the overall output is done as usual.

For a rule with $w < 0$ the conclusion becomes a "negative rectangle" of height $w\tau$ over the whole output domain. After maximum combination with the conclusion fuzzy sets of other possibly active rules, such "negative fuzzy sets" disappear from the output. Rules with negative weights never take part in the overall output value, if maximum combination is implemented by the book and takes *all* fuzzy rules into account – even those with zero activation.
However, we have to be careful, if we use an implementation that skips rules with zero activation or areas with zero membership from the maximum combination for the sake of computation time. In this case, we can obtain a "negative output fuzzy set", or output "fuzzy sets" with negative and positive "membership degrees". It depends on the defuzzification method what kind of crisp output is computed from this. If we use a fuzzy system shell that allows to use negative weights, we must have a close look on how max-min-inference and defuzzification is really implemented.
In general, rules with negative weights only influence the computation of the overall output value, if

maximum combination is implemented incompletely as described above, or if we use local defuzzification methods, or if we have a fuzzy classifier, where the (weighted) degree of fulfilments are combined directly to indicate a class membership, or if we use a Sugeno-type fuzzy system.

• If the conclusion is determined by dot-implication then the output fuzzy sets are rescaled by the weighted degrees of fulfilment before they are accumulated and defuzzified. Now the case $w\tau > 1$ also influences the computation of the output value. If there are negative rule weights, there are mirrored rescaled "membership functions", which disappear after maximum combination, i.e. the effects are similar as those described above for the case on min-implication.

In Sugeno-type fuzzy systems the degree of fulfilment is used to compute a weighted sum. Rule weights occur as factors in the sums of the nominator and denominator:

$$z = \frac{\sum_i \tau_i \cdot w_i \cdot f_i(x,y)}{\sum_i \tau_i \cdot w_i}.$$

We obtain a different result compared to applying weights to the consequents, where the weights would only occur in the nominator (see below).

2. If a rule weight is applied to the consequent part of a rule, it modifies the size of a rule's output value. If the output is crisp, the weight is simply multiplied with this value. If an output fuzzy set is multiplied with a weight, its support and shape is modified. This means we can replace the weights by changing the membership functions in the consequents:

M_1^*: if x is A and y is B then z is D',
M_2^*: if x is A and y is C then z is D''.

Let us assume that the support of the original (convex) output fuzzy set D is $[l, u]$. Then we obtain for the support of D' and D'' $[lw_1, uw_1]$ and $[lw_2, uw_2]$, respectively, or for negative weights $[uw_1, lw_1]$ and $[uw_2, lw_2]$, respectively. This results in the following problems:

• Instead of a single fuzzy set D now two different fuzzy sets D' and D'' are used in our two rules. However, they are labeled with the same linguistic term, i.e. like in Case 1 we again have two different representations of the same linguistic value in our fuzzy system.

• The membership functions of the consequents are shifted away from their original positions, and their supports are rescaled. On the one hand this can result in undesirable small or large supports and on the other hand a fuzzy set can even migrate from a positive part of its domain to a negative part and vice versa. It is also possible, that a fuzzy set completely leaves its previous domain. If we are interested in interpreting the fuzzy systems we are forced to relabel the fuzzy sets.

If a rule weight is used to modify the conclusion of a rule in a Sugeno-type fuzzy system, then the weights appear as additional factors in the nominator of the equation for the overall output value z:

$$z = \frac{\sum_i \tau_i \cdot w_i \cdot f_i(x,y)}{\sum_i \tau_i}.$$

Applying rule weights to consequents is equivalent to replace the function f_i that computes the output of a rule S_i by $w_i f_i$. In this case we do not encounter semantical problems, because the consequents of Sugeno-type rules are usually not interpreted in any way. This means for Sugeno-type fuzzy systems rule weights that are applied to the consequents are a simple way to train the system in a linear fashion.

Rule weights are a very simple way to obtain adaptability for a fuzzy system. Its performance can be easily tuned by applying a learning algorithm to the rule weights, for example a least mean square procedure. This is much simpler to implement, than to specify learning algorithms that modify fuzzy sets directly. However, as we have shown above, rule weights can be equivalently replaced by changing the fuzzy sets used in a fuzzy rule. If this is done, the effects on the rule base become clearly visible, and it is made clear that the interpretation of the fuzzy system is not so easy anymore. In many cases, rule weights alone are not sufficient to enhance the performance of a fuzzy system by learning, because rule weights cannot change the positions of the antecedent membership functions. If this is necessary, then the parameters of the membership functions must also be tuned.

Some of the problems of rule weights can be removed, if the rule weights are normalised. In this case membership degrees not from $[0, 1]$ do not occur, if there are no negative weights. However, this is usually not possible, if rule weights are used mainly to influence the size of the output. Depending on the kind of defuzzification procedure, normalising the rule weights would also effect the size of the output values.

We can view a fuzzy system as a (fuzzy) combination of local models, which are fuzzy sets for Mamdani-type systems, and (usually) linear models for Sugeno-type systems. If a rule weight is applied to the complete rule, then the influence of the local model represented by the consequent is changed for a certain area of the data space. If a rule weight is applied only to the consequent of a rule, then the represented local model itself is changed. For the sake of interpretation these changes should not be done by using rule weights, but instead by changing the respective membership functions directly. If the learning process modifies parameters of the membership functions, it is also easily possible to define restrictions for the learning process [9]. By restricting the learning algorithms we sacrifice degrees of freedom in exchange for readability. Learning becomes more difficult but overfitting might not occur that easily and the learning result can be interpreted more easily.

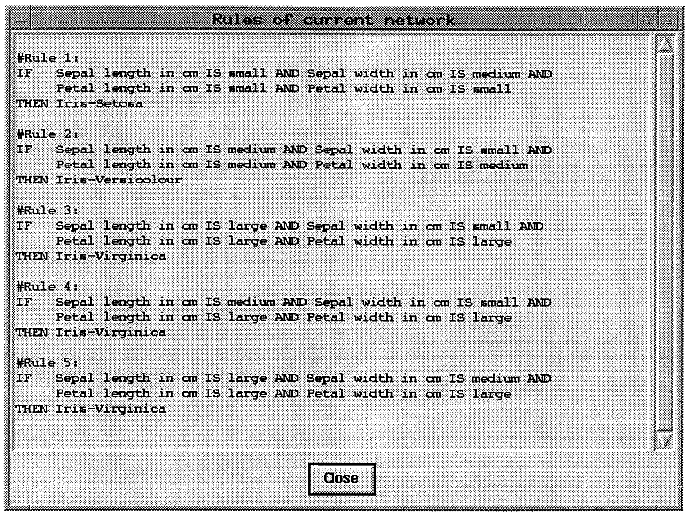
4. A Simple Example

To illustrate the effect of rule weights we use a simple example. We apply our neuro-fuzzy learning environment NEFCLASS [11], [12] to the well-known Iris data set [3]. NEFCLASS is available for download from our homepage at http://fuzzy.cs.uni-magdeburg.de. We chose the Iris example, because it is very simple and therefore suitable for demonstrating the effect of rule weights. The learning result shown here is of course not the best one that can be obtained with NEFCLASS for this data set (compare e.g. [10] or [9]), but it suits our purposes here.

We used three fuzzy sets for each variable and let NEFCLASS find the best 5 rules (for a complete description of the rule learning algorithm of NEFCLASS refer to [9]). We trained the membership functions of the fuzzy classifier in the first trial without rule weights and in the second trial with using additional rule weights. Because our example is from the domain of fuzzy classification, we applied the rule weights to the degrees of fulfilment (Case 1).

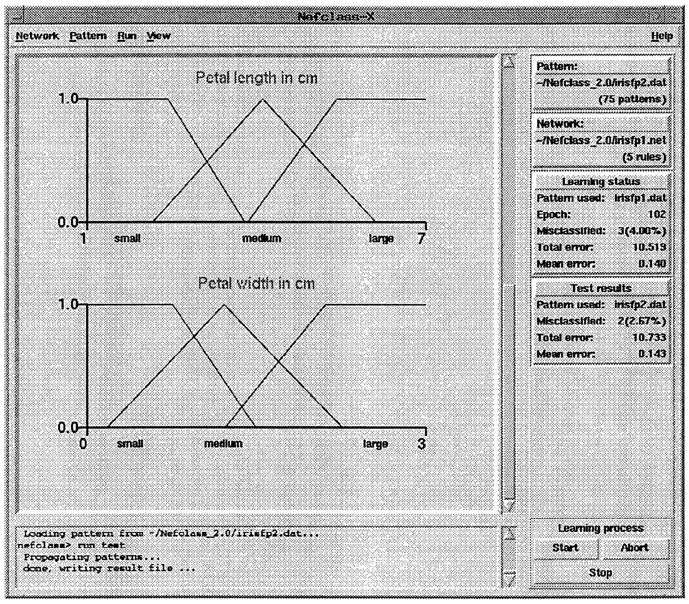
If we learn only rule weights for the five rules, and leave the parameters of the membership functions alone, we obtain an unacceptable result due to unclassified patterns (10 on training set and 10 on test set). If we let NEFCLASS find all rules that are supported by the training set (19 rules for this example) and learn only rule weights, we still obtain 5 errors on the training set and 10 errors on the test set. Therefore we do not consider this configuration further.

The rule base found by NEFCLASS is shown in Figure 1. The learning process was done on one half of the data set using a learning rate of 0.01 for all parameters, and learning was stopped after the error was not decreased further for more than 30 epochs. Testing was done with the second half of the data set. Figures 2 and 3 show the resulting membership functions for the third and fourth variable (petal length and petal width). Figure 3 shows the membership functions before the rule weights are replaced by changing the fuzzy sets, i.e. the weights are still attached to the rules. Replacing the weights is considered below. As you can see



```
#Rule 1:
IF    Sepal length in cm IS small AND Sepal width in cm IS medium AND
      Petal length in cm IS small AND Petal width in cm IS small
THEN Iris-Setosa

#Rule 2:
IF    Sepal length in cm IS medium AND Sepal width in cm IS small AND
      Petal length in cm IS medium AND Petal width in cm IS medium
THEN Iris-Versicolour

#Rule 3:
IF    Sepal length in cm IS large AND Sepal width in cm IS small AND
      Petal length in cm IS large AND Petal width in cm IS large
THEN Iris-Virginica

#Rule 4:
IF    Sepal length in cm IS medium AND Sepal width in cm IS small AND
      Petal length in cm IS large AND Petal width in cm IS large
THEN Iris-Virginica

#Rule 5:
IF    Sepal length in cm IS large AND Sepal width in cm IS medium AND
      Petal length in cm IS large AND Petal width in cm IS large
THEN Iris-Virginica
```

Fig. 1. Five rule to classify the Iris data set obtained from NEFCLASS



```
Pattern:
~/Nefclass_2.0/irisfp2.dat
(75 patterns)
Network:
~/Nefclass_2.0/irisfp1.net
(5 rules)
Learning status
Pattern used:  irisfp1.dat
Epoch:                 102
Misclassified: 3(4.00%)
Total error:        10.519
Mean error:          0.140
Test results
Pattern used:  irisfp2.dat
Misclassified: 2(2.67%)
Total error:        10.733
Mean error:          0.143
```

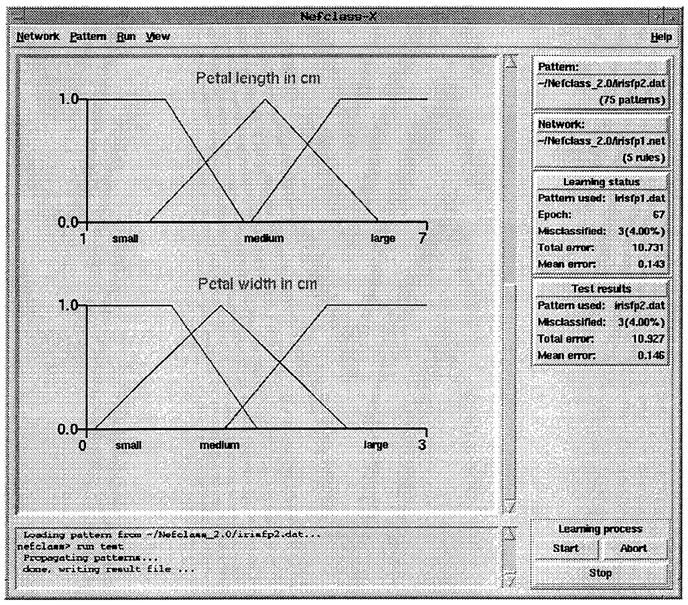
```
 Loading pattern from ~/Nefclass_2.0/irisfp2.dat...
nefclass> run test
 Propagating patterns...
 done, writing result file ...
```

Fig. 2. Trial 1: Resulting fuzzy sets for the third and fourth variable without using rule weights

the fuzzy sets are very similar in both cases. However, the learning results are different as shown in Table I.

The learning result that does not use rule weights is a little bit better than the result that uses rule weights. However, we do not want to consider this fact here, because we just ran the learning procedure once. With other parameters the results might be different. It is only important that the results are similar. We just want to make clear what the effect of the rule weights is in this case.

Let us consider Rules 3 and 4 for the second trial,



Pattern:
~/Nefclass_2.0/irisfp2.dat
(75 patterns)

Network:
~/Nefclass_2.0/irisfp1.net
(5 rules)

Learning status	
Pattern used:	irisfp1.dat
Epoch:	67
Misclassified:	3(4.00%)
Total error:	10.731
Mean error:	0.143

Test results	
Pattern used:	irisfp2.dat
Misclassified:	3(4.00%)
Total error:	10.927
Mean error:	0.146

```
 Loading pattern from ~/Nefclass_2.0/irisfp2.dat...
nefclass> run test
 Propagating patterns...
 done, writing result file ...
```

Fig. 3. Trial 2: Resulting fuzzy sets for the third and fourth variable using rule weights (rule weights not yet replaced)

TABLE I
Learning results for the Iris data set

errors			rule weights				
train.	test	epochs	R_1	R_2	R_3	R_4	R_5
3	2	102	–	–	–	–	–
3	3	67	3.36	4.60	4.04	3.15	3.38

where rule weights are used. Both rules use *petal length is large* in the antecedent. As shown above, we can replace the rule weights by modifying the membership functions of the rules. For Rule 3 we obtain

$$\text{large} : \mathbb{R} \longrightarrow [0, 4.04],$$

and for Rule 4 we have

$$\text{large} : \mathbb{R} \longrightarrow [0, 3.15].$$

This means we have two different interpretations for *petal length is large*, where there is only one interpretation in the first trial, where no rule weights are used. Strictly spoken, *large* is no longer a fuzzy set. Even if we normalize the weights (which would not change the performance of the classifier), we still have the problem of two different non-normal fuzzy sets for the same linguistic term.

As the learning result for the first trial shows, we can obtain an acceptable performance without using rule weights and enjoy the benefit of an interpretable solution.

5. Conclusions

We have shown how rule weights can destroy the linguistic interpretability of fuzzy systems used in function approximation domains like fuzzy classification or fuzzy control. Rule weights can always be replaced equivalently by modifying the membership functions of a fuzzy system. If this is done, the influence of rule weights to the readability of the rule base becomes obvious. Rule weights cause non-normal fuzzy sets and different representations for the same linguistic term

If the linguistic interpretation of a fuzzy system that is modified in a learning process is important, then one should avoid using rule weights. In this case it is better to use a restricted learning procedure that modifies parameters of membership functions, but recognizes constraints that ensure the interpretability of the result obtained by learning. Using our neuro-fuzzy approach NEFCLASS we have demonstrated the effects of rule weights with a simple example, and we have shown that it is not necessary to use rule weights for learning in fuzzy systems.

References

[1] Didier Dubois, Jerome Lang, and Henri Prade. Automated reasoning using possibilistic logic: Semantics, belief revision and variable certainty weights. In *Proc. 5th Workshop on Uncertainty in Artificial Intelligence*, pages 81–87, Ontario, 1989. Windsor.

[2] Didier Dubois and Henri Prade, editors. *Possibility Theory*. Plenum Press, New York, 1988.

[3] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(Part II):179–188, 1936.

[4] Saman K. Halgamuge and Manfred Glesner. Neural networks in designing fuzzy systems for real world applications. *Fuzzy Sets and Systems*, 65:1–12, 1994.

[5] D.E. Heckerman. Probabilistic interpretation for mycin's certainty factors. In J.F. Lemmer and L.N. Kanal, editors, *Uncertainty in Artificial Intelligence (2)*, pages 167–196. North-Holland, Amsterdam, 1988.

[6] Bart Kosko. *Neural Networks and Fuzzy Systems. A Dynamical Systems Approach to Machine Intelligence*. Prentice Hall, Englewood Cliffs, NJ, 1992.

[7] Rudolf Kruse, Jörg Gebhardt, and Frank Klawonn. *Foundations of Fuzzy Systems*. Wiley, Chichester, 1994.

[8] Rudolf Kruse, Erhard Schwecke, and Jochen Heinsohn. *Uncertainty and Vagueness in Knowledge-Based Systems: Numerical Methods*. Springer-Verlag, Berlin, 1991.

[9] Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Foundations of Neuro-Fuzzy Systems*. Wiley, Chichester, 1997.

[10] Detlef Nauck and Rudolf Kruse. A neuro-fuzzy method to learn fuzzy classification rules from data. *Fuzzy Sets and Systems*, 89:277–288, 1997.

[11] Detlef Nauck and Rudolf Kruse. New learning strategies for NEFCLASS. In *Proc. Seventh International Fuzzy Systems Association World Congress IFSA'97*, volume IV, pages 50–55, Prague, 1997.

[12] Detlef Nauck, Ulrike Nauck, and Rudolf Kruse. Generating classification rules with the neuro-fuzzy system NEFCLASS. In *Proc. Biennial Conference of the North American Fuzzy Information Processing Society NAFIPS'96*, Berkeley, CA, June 1996.