Fuzzy Decision Tree FID

Cezary Z. Janikow
Math & Computer Science Department
University of Missouri – St. Louis
St. Louis, MO 63121, USA
janikow@umsl.edu

Krzysztof Kawa
Math & Computer Science Department
University of Missouri – St. Louis
St. Louis, MO 63121, USA
kk7b2@umsl.edu

***Abstract* - FID Fuzzy Decision Tree has been introduced in 1996. It is a classification system, implementing the popular and efficient recursive partitioning technique of decision trees, while combining fuzzy representation and approximate reasoning for dealing with noise and language uncertainty. Since its introduction, various extensions have been proposed, implemented, and presented, with the most recent revision FID3.4 just released. This paper illustrates its processing using a standard example.**

I. INTRODUCTION

Today, computer programs extracting knowledge from data successfully attempt alleviate the knowledge engineering bottleneck problem [1]. Supervised classification programs extract some form of knowledge from training data with known classifications. The knowledge is often in the form of an explicit data structure with some inference method. Among such systems, those building decision trees are very popular, due to their conformity, comprehensibility, accuracy, and low complexity [9][11].

Decision trees implement the so called recursive portioning algorithm, which is a data-driven technique for building tree-based knowledge models [9]. The algorithm starts with all the training data in the root node, and it recursively selects a test, based on the available attributes, to split the node. The selected test is the one that maximizes some measure, such as the information gain [9][11]. The splitting stops based on a number of potential criteria, such as exhausted test attributes or exhausted data samples, and just to avoid overspecialization [11]. The tree can subsequently be visualized to provide meaningful comprehensible information about the domain, or it can be used for classification when coupled with a simple inference procedure – match a new datum against the tree, select the leaf that matches it, and report the decision associated with that leaf.

Neural networks are as attractive for data classification. They provide more robust behavior, but lack similar levels of comprehensibility [2]. A fuzzy approach attempts to bridge the gap between incomprehensible quantitative processing and comprehensible qualitative processing. Fuzzy sets provide basis for fuzzy representation [3]. Fuzzy sets and fuzzy logic allow the modeling of language-related uncertainties, while providing a symbolic framework for knowledge comprehensibility [4][5][6]. In fuzzy rule-based systems, the symbolic rules provide for ease of understanding and/or transfer of high-level knowledge, while the fuzzy sets, along with fuzzy logic and approximate reasoning methods, provide the ability to model fine knowledge details [6]. Thus fuzzy representation provides means for dealing with problems of uncertainty, noise, and inexact data [7]. It has been successfully applied to problems in many industrial areas [8].

Fuzzy Decision Tree (FID) combines fuzzy representation, and its approximate reasoning, with symbolic decision trees. As such, they provide for handling of language related uncertainty, noise, missing or faulty features, robust behavior, while also providing comprehensible knowledge interpretation. FID has three major components: one for partitioning continuous attributes, one for building an explicit tree, and one for knowledge inference from the tree. In this paper, we illustrate these components, along with some additional capabilities.

II. DATA

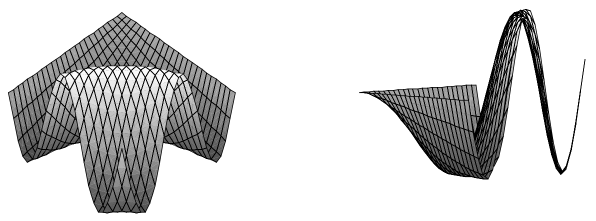


Fig. 1 The original function.

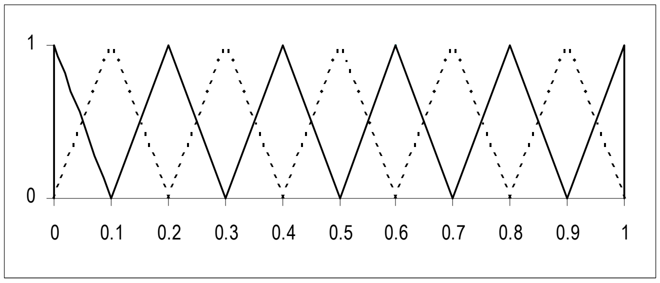


Fig. 2 Partitions of variables X and Y.

For illustration, we are using the function $z = -sin(xy)+1$ taken from [18]. Having a function with only 2 independent attributes allows us to graphically illustrate the acquired knowledge. The X and Y domains are 0..pi. Unless otherwise noted, the independent attributes and the class (Z) are partitioned as illustrated in Fig. 2 (normalized scale). The training is performed with data from the 11 x 11 grid, while the acquired knowledge is generated by sampling the 101x101 grid. Of course, the acquired knowledge depends on both the generated tree and the inference method used for testing – several inferences are illustrated.

III. Fuzzy Sets, Norms, and the Input Language

In classical set theory, given some universe of discourse, an element either belongs to a set or it does not. Such sets can be described with binary characteristic functions. However, in the real world there are many sets for which it is difficult or even impossible to determine if an element belongs to such sets. For example, this information might not be available, or the set itself may be ambiguous due to the linguistic interpretation of its label. Fuzzy sets have been proposed to deal with such cases. A fuzzy set is represented by linguistic label and a membership function onto the real interval [12]. The partitions of Fig. 2 are examples of triangular fuzzy sets. In the figure, an element with the normalized value 0.05 would have memberships in both two leftmost sets equal to ½.

Similar to the basic operations of union, intersection, and complement defined in classical set theory (which can all be defined with the characteristic function only), such operations are defined for fuzzy sets. Functions used to interpret intersection are denoted as T-norms, and functions for union are denoted as S-norms. Originally proposed by Zadeh, *min* and *max* operators for intersection and union, respectively, define the most optimistic and the most pessimistic norms for the two cases.

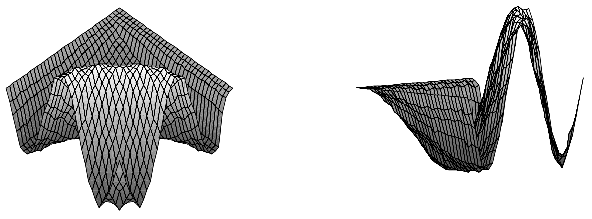


Fig. 3 Response curve under the minimum norm

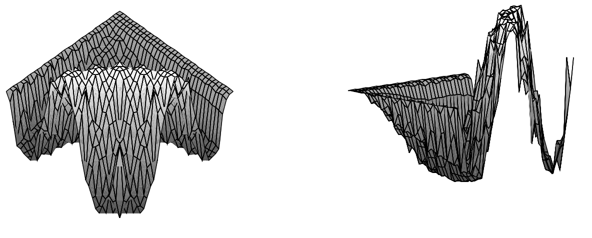


Fig. 4 Response curve under the bounded product norm

A decision tree has some attribute tested in each of its internal nodes, and has classification decisions associated with the leaves. The test in a given node is interpreted as a fuzzy restriction, and its satisfaction is computed based on the attribute value of a sample. When moving down a path, the degrees of satisfaction of the fuzzy restrictions are accumulated according to the fuzzy T-norms. In FID, minimum, product, bounded product and drastic product can be selected. There is also the 'best' option, which chooses the best of the above. Fig. 3 and Fig. 4 illustrate the difference between two different T-norms.

FID processes data expressed with a mixture of nominal and continuous attributes. The continuous attributes can be partitioned or not, and some features can be missing. For example, assuming attributes *Income* (partitioned, with linguistic labels *Low*, *Medium*, *High*), *Employment* (expressed as the number of hours per week), nominal *Sex*, and classification for *Credit* (score 0-100, partitioned with three labels *No*, *Maybe*, *Yes*), the following are examples of possible training data:

John: *Income=12500*, *Employment*=24, *Sex=M*, *Credit=No*
Susan: *Income=Medium*, *Employment*=?, *Sex=F*, *Credit*=25

IV. Inferences

The tree is built for the training data, and it can be easily interpreted. However, to classify new data, an inference mechanism is needed, one which uses elements of the tree to classify the data. FID3.4 uses two different kinds of inferences: set-based and exemplar-based. The set-based inferences generate fuzzy sets in the leaves, following composition along the path from the root. Exemplar-based inferences treat each leaf as a super-exemplar. Such exemplar can be the most dominant training datum in the leaf, or a weighted center of gravity of the data in there, etc. [12][16].

When a new datum is being classified, it is first matched against the tree, using the internal tests, the associated fuzzy restrictions, and the T-norms. Because of the nature of the fuzzy sets, more than one leaf can get activated, with different degree of match. The multiple leaf activations are the so called external conflicts. In addition, each leaf may contain training data from multiple classes. These are the so called internal conflicts. A given inference must specify how to resolve all these conflicts.

In the set-based inferences, there are four different external, and four different internal, conflict resolutions:

1. External resolutions:
   - Average over multiply matched leaves.
   - Do not average over multiple leaves (just operate on individual classes).
   - Use only single leaf, the one with the highest match to the datum.
   - Max sum gravity.
2. Internal resolutions:
   - **best**: use only the best majority class in the leaf.
   - **cg**: computes center of gravity of the entire leaf.
   - **mcg**: max-center of gravity method.
   - **all**: use all decision classes in the leaf.

In the exemplar-based inferences, there are three different external, and four different internal, conflict resolutions:

1. External resolutions:

- Use all the matched leaves as exemplars.
- Look for support for individual decisions throughout all matched leaves.
- Use only the best-matched leaf is the exemplar.

2. Internal resolutions:
   - **all**: create an exemplar over all data in the leaf.
   - **best**: create an exemplar over only the data of the best class in the leaf.
   - **maxx**: use the training datum with the highest match in the leaf as the exemplar for the leaf.
   - **maxxk**: same as above but consider only data from the majority class in the leaf.

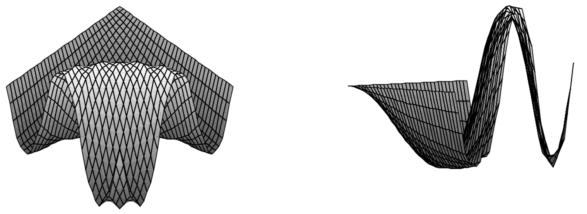


Fig. 5 Set-based inference. External: average over multiple leaves. Internal: center of gravity of a leaf.

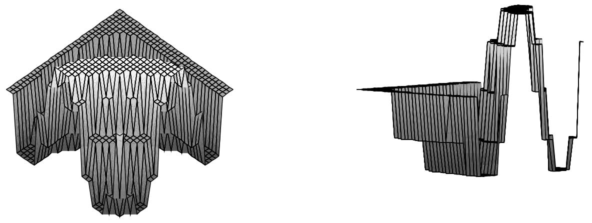


Fig. 6 Set-based inference. External: multiple leaves not averaged. Internal: all decision classes in a leaf

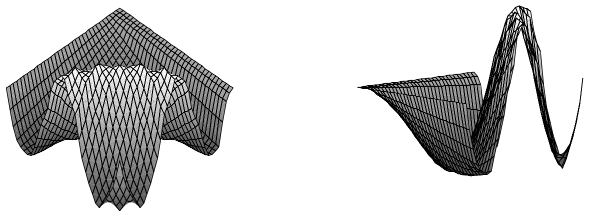


Fig. 7 External: all leaves are exemplars. Internal: exemplar over all examples

Note that not every internal resolution method can be used for a given choice of the external resolution. Illustrations of some responses from various inferences are shown in Fig. 5-8. In all cases the product T-norm was used. As seen, there are some differences in the degree of detail. The differences are much more profound if the data is noisy or incomplete, as seen in the following sections.

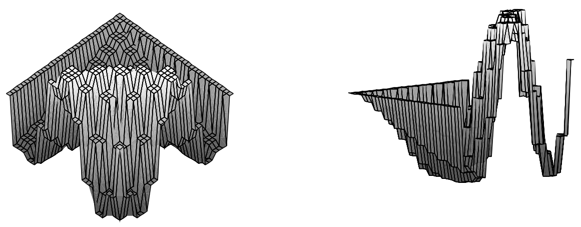


Fig. 8 External: use only the best leaf. Internal: exemplar over all examples

V. MISSING VALUES

In practice, attribute features can often be missing. To deal with missing values, we borrow the existing decision tree methodology [10]. The best approach is to evenly split an example into all children if the needed feature value is not available, and then to reduce the attribute's utilization by the percentage of examples with unknown value. We performed two tests, with random removal of 10% and 50% of the features from the training sets.

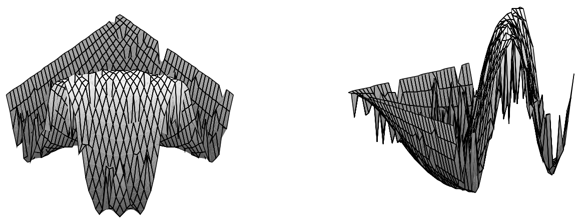


Fig. 9 Response curve when trained with 10% of missing values, using set-based inference.

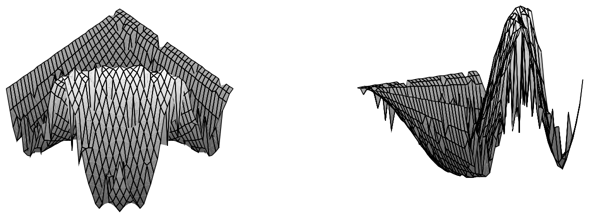


Fig. 10 Response curve when trained with 10% of missing values, using exemplar-based inference.

Fig. 9 and 10 show the responses generated from the tree when trained with 10% of missing features while using set-based and exemplar-based inferences (those illustrated in Fig. 5 and Fig. 7, respectively). Fig. 11 and Fig. 12 show responses generated from the tree when trained with 50% of missing features, while using the same two inferences as above. As seen, even though the overall function shape is preserved, there are some tests with high error (higher in the 50% case). One obvious explanation for the errors is that the training was done on the 11x11 grid, which meant one data point per fuzzy set of the dependent attributes. When values on neighboring fuzzy sets are missing, this causes quite a few leaves generated without actual data values. When there is more data per partitioned set, the results should be even better.

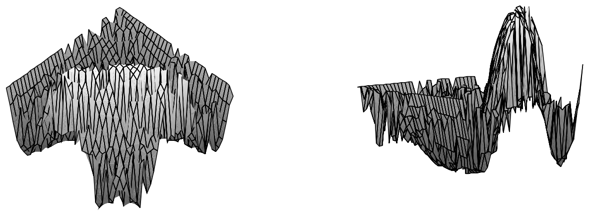


Fig. 11 Response curve when trained with 50% of missing values, using set-based inference.

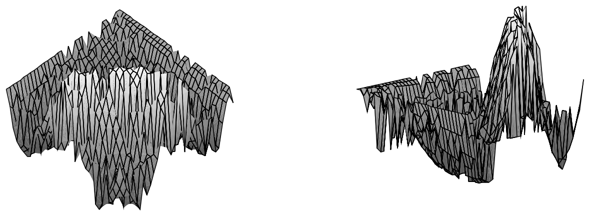


Fig. 12 Response curve when trained with 50% of missing values, using exemplar-based inference.

VI. NOISE

In practice, attribute values (features) can also be incorrect, resulting from faulty sensors. There are two different cases here. First, if the sensor is completely faulty, it would result in random measurements. Second, the sensor can be just inaccurate, resulting in some noise superimposed on the actual feature value. To illustrate FID's behavior under such conditions, we assumed the first case. Accordingly, we performed two tests, randomly replacing 10% and 50% of the available features in the training data. The noise was uniformly distributed in the domain.

First, to illustrate the effect of such random noise on the function, we illustrate the actual function with the same amount of noise added. The resulting function is shown in Fig. 13.

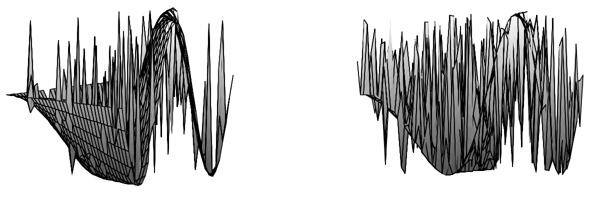


Fig. 13 The actual function generated with 10% (left) and 50% (right) of random noise.

As seen, especially with the higher noise the function becomes visually lost.

Subsequently, we trained FID with such noisy data and performed testing of the acquired knowledge when interpreted by two different inferences. Fig. 14 shows the response generated by the set-based inferences as in Fig. 5. Fig. 15 shows the response generated by the exemplar-based inferences as in Fig. 7. As seen, both inferences are able to filter-out the noisy information from the tree. For the used conflict resolutions, the exemplar-based inferences seem to perform better especially for the 50% case.

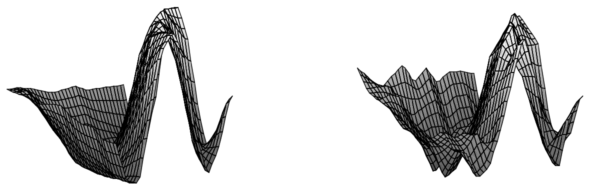


Fig. 14 Response curve on noisy data, from the set-based inference (10%-left, 50%-right)

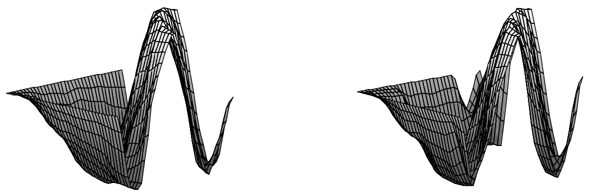


Fig. 15 Response curve on noisy data, from the exemplar-based inference (10%-left, 50%-right)

VII. PARTITIONING

When the universe of discourse is not partitioned, it will have to be discretized prior to or during tree building. FID can perform two kinds of discretizations: local top-down, and global bottom-up [14]. Both methods are data driven, and only discretize the attributes that need to be partitioned. The top-down method starts with a single partition (that is, no partitioning) assigned to each applicable attribute, and then partitions only the attribute that when partitioned it improves the information measure in the node. This is performed while building the tree. However, this procedure is not straight depth-first traversal, but instead the nodes are placed on a priority queue based on the number of data they contain. The reason is that each node splitting can result in a new partition added (if this improves the information measure), and such partitions should be generated based on the maximal amount of information available (the number of training data). As a result, only the most relevant attributes are partitioned.

There is a limit on the maximal number of partitions allowed on an attribute.

The bottom-up method works in opposite [14] and has opposite properties. It starts with each attribute partitioned to sets assigned to individual data points. These detailed partitions are later merged, on all attributes being partitioned. As a result, all attributes get portioned, but the method is less efficient for larger number of data and attributes. There are limits on both the minimal and the maximal number of partitions per attribute.

Fig. 16 shows the resulting partitioning of the X attribute generated by the top-level method, while Fig. 17 shows the same generated by the bottom-up method. Minimum partitions were set to 1, maximum to 11 (this was performed with more training data to avoid partitioning into one set per data point). As seen, both methods generated denser partitions in the subdomain where the function changes values most rapidly. However, the bottom-up method clearly does it more greedily (this can be afforded as the second attribute, Y, was already partitioned as in Fig. 2, and thus partitioning of X could indeed be more relaxed).

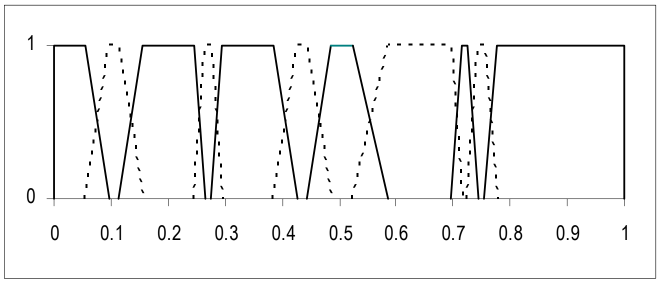


Fig. 16 Top-down partitioning on X.

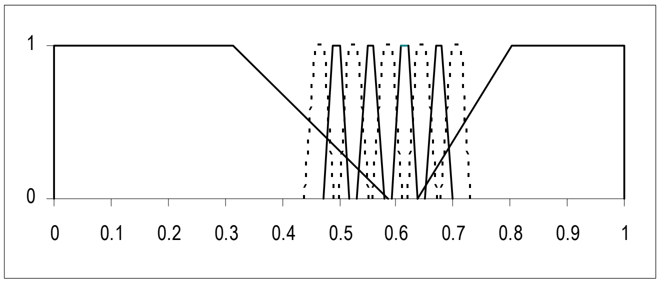


Fig. 17 Bottom-up partitioning on X.

Subsequently, we performed top-down partitioning of both attributes simultaneously. Fig. 18 presents the resulting sets. The generated sets are always trapezoidal. However, the shape of the trapezoid is controlled by a parameter. For example, in the top-down partitioning, setting the parameter to 0 results in triangular partitions. Two different results are illustrated in Fig. 19 (one internal rule forces the sets to always intersect at 0.5, another one allows only neighboring sets to intersect).

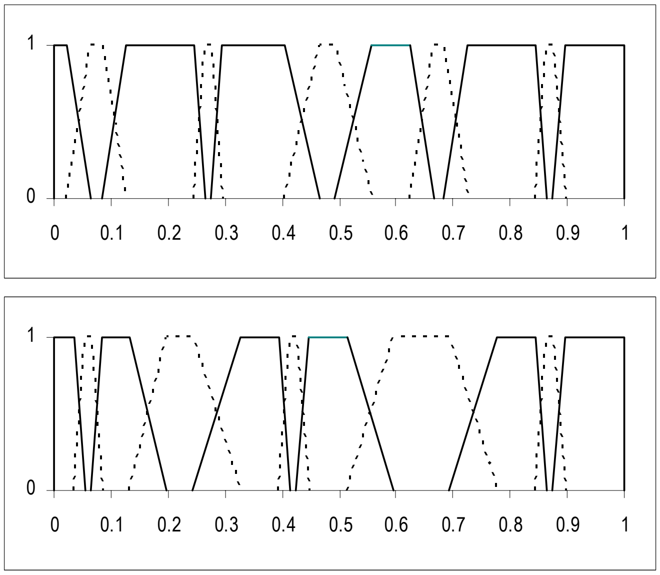
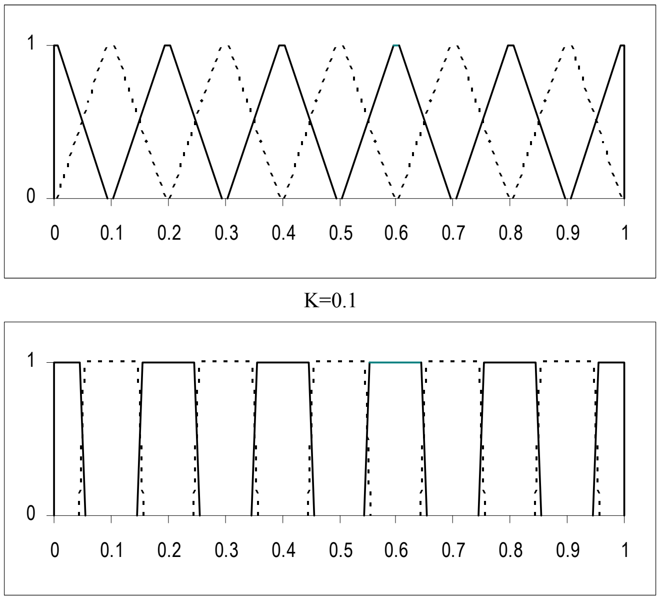


Fig. 18 Top-down partitioning on X and Y.



K=0.9

Fig. 19 Different results of partitioning for two different parameter K values.

VIII. Pruning

FID3.4 contains pruning algorithm to prevent overfitting, which balances a tradeoff between the size of the tree and its predictive accuracy. Pruning is based on a method proposed by Quinlan in the C4.5 system [11] and modified to utilize fuzzy logic. The only parameter used for pruning is a confidence level (CF).

The complete tree built from this training data set had 111 leaves. The response from an un-pruned tree, using the set-based inference with the averaging external and best internal conflict resolutions, is illustrated in Fig. 20. Note that the result is less smooth as compared to that in Fig. 5 – the result of a less robust inference method (internal best here).

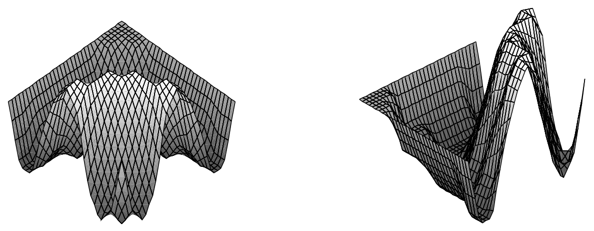


Fig. 20 Response curve from un-pruned tree.

Subsequently, we pruned the tree to reduce the number of leaves. The resulting response became less and less robust, as illustrated in Fig. 21 and 22.

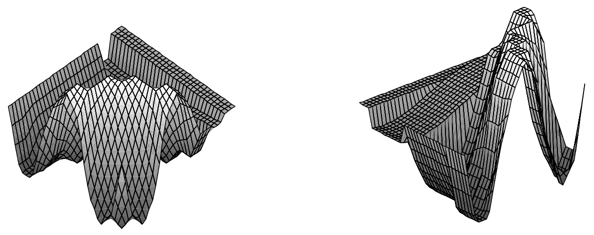


Fig. 21 Pruning; CF=0.2; 61 leaves

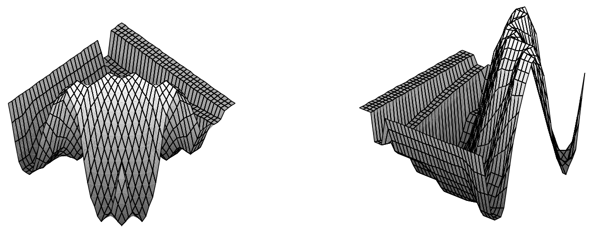


Fig. 22 Pruning; CF=0.5; 91 leaves

IX. Summary

We have illustrated the FID3.4 classification system here, just recently released. The system is a decision tree, utilizing fuzzy logic, approximate reasoning methods, as well as methods from machine learning. In particular, the system can handle data expressed in a mixed language of nominal, fuzzy, and continuous domains. It uses a number of T-norms, and a number of inferences, based on approximate reasoning and on exemplar-based learning. Due to the richness of the information processed by some of the inferences, the system handles missing features and noise quite robustly, as illustrated.

Finally, FID3.4 includes domain partitioning for continuous attributes, as it does a pruning method.

Other options, not illustrated in the paper, include tests for attribute relevancy, thresholds for classifications in leaves to be considered in the inferences, gradually relaxed classification means if a test datum is not classified, etc. More information can be found at the User's Manual, available from `http://www.cs.umsl.edu/~janikow/fid.`

X. Bibliography

[1] R.S. Michalski, "Understanding the nature of learning", In Machine Learning: An Artificial Intelligence Approach, R. Michalski, J. Carbonell & T. Mitchell (eds.), Vol, II, pp. 3-26. Morgan Kaufmann, 1986

[2] S. Sestino & T. Dillon, "Using single-layered neural networks for the extraction of conjunctive rules and hierarchical classifications", Journal of Applied Intelligence 1, pp. 157- 173, 1991.

[3] L.A. Zadeh, "Fuzzy sets", Information and Control 8 (1965), pp. 338-353.

[4] L.A. Zadeh, "Fuzzy logic and approximate reasoning", Synthese 30 (1975), pp. 407-428.

[5] L.A. Zadeh, "A theory of approximate teasoning", In Hayes, Michie & Mikulich (eds) Machine Intelligence 9 (1979), pp. 149-194.

[6] L.A. Zadeh, "The role of fuzzy logic in the management of uncertainity in expert systems", Fuzzy Sets and Systems, 11, 1983, pp. 199-227.

[7] D. McNeill & P. Freiberger, "Fuzzy logic", Simon & Schuster, 1993.

[8] A. Kandel & G. Langholz (eds.), "Fuzzy control systems", CRC, 1993.

[9] J.R. Quinlan, "Induction on decision trees", Machine Learning, Vol. 1, 1986, pp. 81-106.

[10] J.R. Quinlan, "Unknown attribute-values in induction", In Proceedings of the Sixth International Workshop on Machine Learning, 1989, pp. 164-168.

[11] J.R. Quinlan, C4.5: "Programs for machine learning", Morgan Kaufmann, San Mateo, CA. 1993.

[12] C.Z. Janikow, "Fuzzy decision trees: issues and methods", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 28, Issue 1, pp. 1-14. 1998.

[13] C.Z. Janikow, M. Fajfer, "Fuzzy partitioning with FID3.1", Proceedings of the 18th International Conference of the North American Fuzzy Information Society, NY 1999, pp. 467-471.

[14] M. Fajfer, C.Z. Janikow, "Bottom-up partitioning in fuzzy decision trees", Proceedings of the 19th International Conference of the North American Fuzzy Information Society, Atlanta 2000, pp. 326-330.

[15] Cezary Z. Janikow, "Fuzzy decision forest", Proceedings of 22nd International Conference of the North American Fuzzy Information Processing Society, Chicago 2003, pp. 480-483.

[16] Cezary Z. Janikow, "FID4.1: an overview", Proceedings of NAFIPS 2004, pp. 877-881.

[17] J.R. Quinlan, "The effect of noise on concept learning", Machine Learning II, R. Michalski, J. Carbonell & T. Mitchell (eds), Morgan Kaufmann, 1986.

[18] J. F. Baldwin and Dong (Walter) Xie, "Simple fuzzy logic rules based on fuzzy decision tree for classification and prediction problem", Intelligent Information Processing (IIP) II, Beijing, China , pp 175-184, 2004.