FID4.1: an Overview

C.Z. Janikow

Department of Mathematics and Computer Science
University of Missouri – St. Louis
St. Louis, MO 63121 USA

cjanikow@ola.cs.umsl.edu

***Abstract* – FID4.1 is a freeware software tool for supervised classification, using fuzzy decision tree and forest. It is loosely based on the ID3 decision tree algorithm. It handles nominal, continuous, and linguistic attributes and classes. It can also operate with noisy and unknown features, both in training and testing data. For continuous attributes that are not partitioned into fuzzy sets, the system generates fuzzy partitioning using top-down and bottom-up methods. Finally, FID uses a number of inferences, falling into two classes: set-based and exemplar-based. In this paper we overview the major features and functionalities of FID4.1.**

I. Introduction

In the mass storage era, knowledge acquisition represents a major knowledge engineering bottleneck. Computer programs extracting knowledge from data successfully attempt to alleviate this problem. Supervised classification systems are computer programs which extract some form of knowledge from data represented by training data with known classifications. The knowledge is often in the form of explicit data structure plus an inference method. Among such systems, those building decision trees are the most popular, due to their conformity, comprehensibility, accuracy, and low complexity.

Decision trees were popularized by Quinlan with the ID3 program [11]. Decision trees are examples of recursive partitioning methods, which are data-driven algorithms building knowledge models in the form of a tree. In the recursive procedure, the algorithm selects a test which maximizes information gain, and then partitions the domain, and the training examples, using the test. In ID3, the test is based on symbolic attribute values [11], while in CART it is based on a threshold [2]. The partitioning stops based on a number of potential criteria – either when no more tests are available, and just to avoid overspecialization. The tree can be used for data classification when coupled with an inference procedure – match a new datum against the tree, select the leaf that "recognizes" it, and report the decision associated with that leaf. Problems rise when no such leaf can be found, multiple leaves are found, in addition with discontinuity of the decision with small variations in data [6][12][13].

Systems based on this approach have traditionally worked well in symbolic domains. More recent extensions are meant to allow operating with continuous attributes, incorporate probabilistic processing, etc. [12][13][14]

In recent years, neural networks have become equally popular due to relative ease of application and abilities to provide gradual responses. However, they generally lack similar levels of comprehensibility. A fuzzy approach attempts to bridge the gap between incomprehensible quantitative processing and comprehensible qualitative processing. Fuzzy sets provide bases for fuzzy representation [15]. Fuzzy sets and fuzzy logic allow the modeling of language-related uncertainties, while providing a symbolic framework for knowledge comprehensibility [16][17]. 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. Accordingly, fuzzy representation is becoming increasingly popular in dealing with problems of uncertainty, noise, and inexact data. It has been successfully applied to problems in many industrial areas [3].

Fuzzy Decision Tree (FID) successfully merged fuzzy representation, with its approximate reasoning capabilities, and symbolic decision trees, while preserving advantages of both: uncertainty handling and gradual processing of the former with the comprehensibility, popularity, and ease of application of the latter [6]. As a decision tree, FID has three major components: one for partitioning any continuous attribute without predefined fuzzy sets, one for building an explicit tree or forest, and one for knowledge inference from the explicit representation. In this paper, we overview the major features and functionalities of FID 4.1, the latest system release.

II. FID History

FID2.0 has been released in 1997. It provided the tree building module as well as an inference module, with approximate reasoning inferences based on fuzzy sets as well as exemplar based inferences. Among additional features were abilities to process missing features, data representation using categorical and fuzzy terms [*], chi-square test to prune insignificant attributes, IV method to alleviate ID's bias toward larger domains, and extended inferences to trigger when the tree fails to classify a new datum.

FID3.2 has been released in 1998. Major changes vs. release v2.0 include explicit processing of categorical features, along with special conflict resolutions, implicit normalization of attributes, as well as a module for partitioning continuous attributes without predefined fuzzy sets.

[*] In FID2.0, categorical domains were processed as non-overlapping fuzzy sets.

FID4.0 has been released in 2000. It major feature is that the decision tree retains alternative split tests, thus creates a 3-D tree (forest). FID4.1, currently in Beta testing, corrects minor errors and resolves platform compatibility problems. We are also working a Java-based platform independent interface for setting run parameters as well as tree visualization.

III. FID FEATURES

FID4.1is a classification system which acquires knowledge in the form of a tree (or forest) and an inference method, based on training data. It can subsequently be used for classification of a new unknown event. FID processes data expressed in terms of: numerical domain values (for continuous or linear numerical attributes), such as *Salary*=45,000; fuzzy terms, such as *Salary* is *High*, and categorical terms, such as *BloodType* is *A*. This applies to both attributes and classification. However, attribute features can be missing, while decision value is required. In addition, training data can be weighted to express relevance of each individual example.

For continuous/linear attributes, the feature can thus be expressed using either the actual domain value, or a fuzzy term corresponding to a fuzzy set from the predefined fuzzy partitioning. However, if the attribute doesn't have the predefined partitioning, all features must be the exact domain values (or unknown). In this case, one of the two partitioning methods will generate fuzzy sets.

A. Domain Partitioning

FID4.1 provides two different partitioning methods: top-down and bottom up. Both are data driven, but the former one partitions the domains, while the latter merges sub-partitions.

The top-down partitioning method works similarly to the CART system [2]. It follows the same recursive partitioning algorithm. However, it is not a depth-first algorithm. Instead, tree nodes (starting with the root node containing all the training data) are placed on a priority queue, ordered by the number of examples contained. The priority technique guarantees that each new partition will be based on the maximum number of training data. The test in a node is the one maximizing the information gain, based on: testing the known attributes; testing the attributes being partitioned while using the fuzzy covering generated so far (a single fuzzy set with complete covering to start with), testing the same attributes but with splitting each of the current fuzzy sets into two overlapping sets. However, each attribute being partitioned is not partitioned any more when a maximal number of fuzzy sets are generated. This recursive procedure stops based on the same stopping criteria as those for building a decision tree. It is fully described in [7].

It is important to note that FID then follows with its normal operation rather than using the generated tree (tree further generated tree will be wider but shallower in general). Also, this technique partitions only those attributes which are the most relevant for information gain, while it fails to partition the other attributes. Therefore, it is less appropriate for the forest, where alternative tests are sought [8][9]. Moreover, this is a local technique, as each new partition is decided based on some local training data contained in a single tree node.

The bottom-up partitioning is a data-driven technique which partitions all domains without predefined partitioning, while retaining the other domains. It operates opposite to the top-down technique, as it starts with maximal partitioning and merges individual partitions. Moreover, it is a global method as it always uses all the available data. Therefore, it is generally preferable, especially for the forest.

The algorithm starts with complete partitioning of the appropriate domains: each attribute-value spans its own partition. These partitions are subsequently merged, using some heuristics following entropy. Each domain is partitioned to a number of fuzzy sets within a user-defined range. The procedure is fully described in [4].

B. Tree Building

The next step is to build the decision tree. While building the tree, the best test (maximizing the information gain) must be selected. The recursive process stops when there are no more available tests, or when information level or the number of remaining examples falls below certain thresholds. When selecting the best test (highest information gain), attributes can be pruned so that FID disregards attributes failing the chi-square test of relevancy. Moreover, attributes with high domain cardinality (which have strong bias in the information gain formula) can have their gain normalized.

To decide on the test, each training example must be matched against the specific test leading to a subtree. If the test is based on a categorical attribute, the outcome of the test is a Boolean value, and thus an example either belongs to the subtree or it doesn't. For other attributes, the test is based on the degree of match of the example's appropriate feature to a fuzzy restriction associated with a fuzzy linguistic value (and thus a fuzzy set). If the example's feature is a numerical domain value, FID uses the membership function as the degree of the match. If the feature is a fuzzy term, FID uses the degree of match between two fuzzy sets [6]. If the feature value is unknown, the example can be disregarded, or alternatively it can be assigned to some of the subtrees.

When moving down the tree, the degrees of satisfaction of the fuzzy restrictions (or the categorical value) are accumulated according to fuzzy T-norms [3][16][17]. The available global options include: *min*, *product*, *drastic product*, *bounded product*, and *best* (locally best).

In FID4.1, each node can select more than one alternative test. For example, two independent tests can produce comparable information gains, and it would be a loss to toss one away. They are controlled by a few parameters, but generally alternatives with similar level of gain and higher in the tree are preferred.

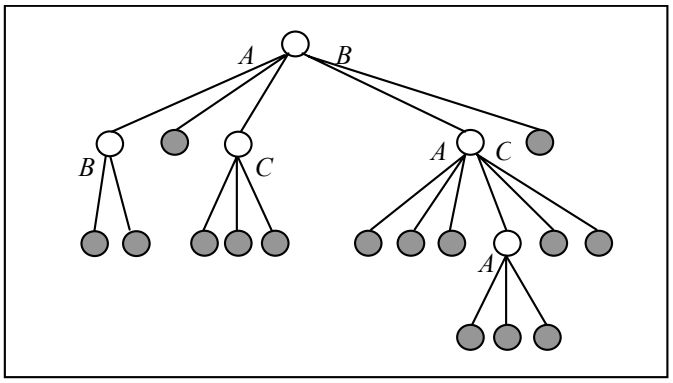

Fig. 1 A sample forest: two tests at the root, and two tests in the right tree slice.

A sample decision forest is presented in Fig. 1. It is created by retaining two alternative tests in the root (based on attribute *A* and *B*). When we consider only one test at a time, we are talking of alternative slices of the forest. Each slice is a forest. However, if one takes only one alternative for every test, a simple tree is obtained. Fig. 2 presents the three possible slice-trees of Fig. 1.

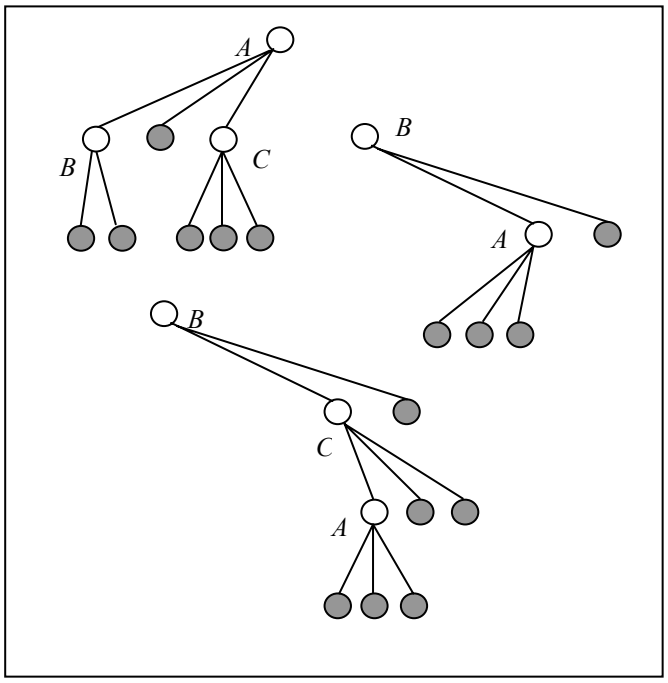

Fig. 2 The three tree slices of the forest in Fig. 1.

A. Knowledge Inferences

An ideal tree will have unique classification of training examples falling into individual leaves. However, the FID tree building procedure can stop node expansion for a number of reasons. Moreover, examples fall into leaves with different degrees (depending on the combined matches to the fuzzy restrictions leading to those nodes). Finally, the example's classification can also be expressed with a numerical value, potentially matching more than one linguistic class value. Therefore, the FID tree is very unlikely to be such an ideal tree. This is in fact the source of the additional information expressed and retained in the tree, and thus should not be discarded.

When a new unknown example is presented for classification, a few tasks take place. First, the example must be matched against the leaves of the tree, and then information inferred from all the leaves must be combined. This processing of the leaves separately follows the idea of local inference rule in fuzzy-based systems [3].

The unknown example is matched against all the fuzzy restrictions leading to all the leaves, one feature at a time. Based on the degree of each match, and on the used T-norms to combine the matches (the same alternatives as for building the tree), the example may end up in a number of different leaves. If a specific feature is unknown, the example is sent to all possible subtrees with reduced degree. If the feature is know, but a subtree is missing (e.g., when there was no training data with this feature), the available feature is fuzzified so that it matches the other domain values (this is done differently for categorical and other attributes) [6]. If the tree is a forest, the example progresses through all the available slices.

When the matching leaves are identified, along with their degree of match, the next step is to take classification information generated by each such leaf individually (resolve so called internal conflict) and the combine information from multiple leaves and slices (resolve so called external conflict).

These conflicts are resolved differently based on the inference method used: approximate-reasoning based on fuzzy sets, or exemplar-based.

Exemplar-based learning refers to acquiring most important exemplar training data (either actual training data or artificially generated example) [1][5][14]. In our case, each individual leaf can be treated as an exemplar. Its classification can be either that most prominent in the leaf, or an aggregated class value [5]. Subsequently, classification from all leaves recognizing the unknown example must be aggregated. This can be done by computing super-exemplar and taking its classification, or by taking the information from the [5].

Approximate-reasoning set-based inferences aggregate the information from multiple leaves following the local-inference rule. The processing can be simplified by representing fuzzy sets by their center of gravity, resulting in center-of-gravity methods (which can additionally be weighted). Alternatively, the fuzzy sets can be processed explicitly with sum-center of gravity or max-center of gravity [3][6]. Moreover, when taking a single leaf into account, one may consider only the fuzzy sets with most training data, or all the fuzzy sets. As an illustration, consider the well known Mexican Sombrero function, plotted in Fig. 3. When acquired with a single decision tree (not a forest), and tested using two different inferences, the function is recovered with different degree of detail, as indicated in Fig. 4.

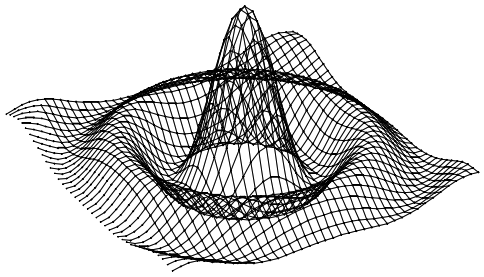

Fig. 3 The Mexican Sombrero function.

Finally, an inference may use just the most prominent tree slice (following the highest-gain test in every node), or it can combine information from leaves of multiple trees, in a similar way as combining information from leaves of a single tree.

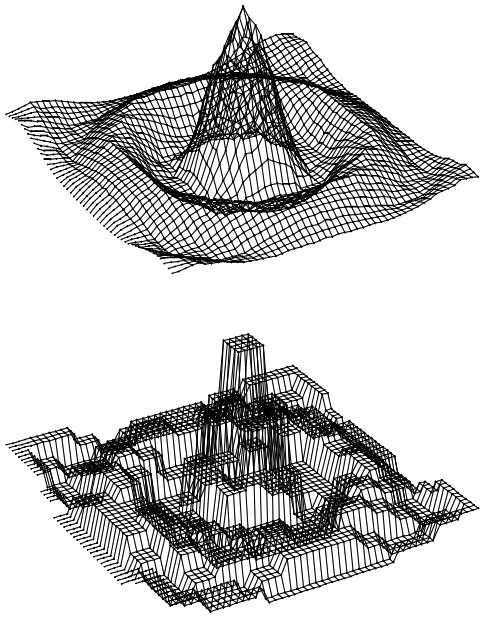

Fig. 4 Visualization of the FID acquired knowledge (tree +inference) for the same tree but two different inferences.

A. Other Features

FID4.1 supports a number of other small features. As previously mentioned, attributes can be pruned for relevancy based on the chi-square test. Bias in attributes with large set of linguistic/nominal values can be normalized. Known attribute values can be processed both during training and testing (gain for attributes with large number of missing features is also reduced following [13]. One feature still remains to be explained.

What happens when a new example fails to be recognized by a given inference. One option would be to trigger a more relaxed inference (e.g., taking more information into account, such as considering a forest rather than a single tree). However, this is left for the user to decide to change the inference. At present, FID can be forced to classify an example using the same inference. This is accomplished by fuzzifying the training example that is replacing its features with newly constructed fuzzy sets (for non-categorical attributes) and relaxing the categorical features, through a few stages, until some matches are found.

REFERENCES

[1] E.R. Baires, B.W. Porter and C.C. Wier. "PROTOS: An Exemplar-Based Learning Apprentice". *Machine Learning III*. Morgan Kaufmann, pp. 112-127.

[2] L. Breiman, J.H. Friedman, R.A. Olsen & C.J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[3] R. Jager. *Fuzzy Logic in Control*. Ph.D. dissertation, Technische Universiteit Delft, 1995.

[4] M. Fajfer and C.Z. Janikow. "Bottom-up Partitioning in Fuzzy Decision Trees". *International Conference of the North American Fuzzy Information Society*, Atlanta 2000 , pp. 326-330.

[5] C.Z. Janikow. "Exemplar Learning in Fuzzy Decision Trees". *Proceedings of FUZZ-IEEE 1996*, pp. 1500-1505.

[6] C.Z. Janikow. "Fuzzy Decision Trees: Issues and Methods". *IEEE Transactions on Man, Systems, Cybernetics*, Vol. 28, Issue 1, pp. 1-14, 1998.

[7] C.Z. Janikow and Maciej Fajfer. "Fuzzy Partitioning with FID3.1". *Proceedings of the 18th International Conference of the North American Fuzzy Information Society*, IEEE 1999, pp. 467-471.

[8] C.Z. Janikow and Maciej Fajfer. "Fuzzy Decision Forest". *International Conference of the North American Fuzzy Information Society, Atlanta 2000*, pp. 218-221.

[9] C.Z. Janikow. "Fuzzy Decision Forest". Proceedings of 22nd International Conference of the North American Fuzzy Information Processing Society, Chicago 2003, pp. 480-483.

[10] R.S. Michalski. "Theory and Methodology of Inductive Learning". *Machine Learning I*, Morgan Kaufmann, 1986, pp. 83-134.

[11] J.R. Quinlan. "Induction on Decision Trees". *Machine Learning*, Vol. 1, 1986, pp. 81-106.

[12] J.R. Quinlan. "Decision Trees as Probabilistic Classifiers". *Proceedings of the Fourth International Workshop on Machine Learning*, 1987, pp. 31-37.

[13] J.R. Quinlan. C4.*5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA. 1993.

[14] J.R. Quinlan. "Combining Instance-Based and Model-Based Learning". *Proceedings of International Conference on Machine Learning 1993*, Morgan Kaufmann 1993, pp. 236-243..

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

[16] L.A. Zadeh. "Fuzzy Logic and Approximate Reasoning". *Synthese* 30 (1975), pp. 407-428.

[17] 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.