Fuzzy Partitioning with FID3.1

Cezary Z. Janikow
Dept. of Mathematics and Computer Science
University of Missouri – St. Louis
St. Louis, Missouri 63121
janikow@umsl.edu

Maciej Fajfer
Institute of Computing SCience
Poznan Institute of Technology
kme@man.poznan.pl

Abstract

FID3.1 builds fuzzy decision trees, with a range of choices for fuzzy operators and inferences. Various FID algorithms are being widely used for dealing with numeric and/or imprecise data, for fuzzy classification or for generating fuzzy rules. FID 3.0 adds a number of new features, the most important being a fuzzy partitioning mechanism - construction of fuzzy sets for continuous variables w/o predefined fuzzy terms. FID3.1 improves the mechanism in a number of ways. This paper describes the partitioning method and presents a few comparative experiments.

1. Introduction

Decision trees are one of the most popular methods for learning and reasoning from feature-based examples [6]. However, they often have been criticized for their persistent over-reliance on near-perfect data, and for the resulting degradation in the presence of imperfect data. Data imperfection might have been the result of noise, imprecise measurements, subjective evaluations, inadequate descriptive language, or simply missing data. Additional problems arise from continuous or simply large nominal attributes - all such domains have to be partitioned. Some of these potential problems have been successfully addressed in the past. For example, Quinlan has proposed some methods for dealing missing features both in training data and in the examples to be classified [7]. Continuous domains have been addressed by CART [1] and subsequently by C4.5 [8] algorithms, with the usual approach being to use threshold values to split a domain. However, the resulting knowledge generally exhibits lower comprehensibility and overspecialization. These problems were in turn addressed by tree pruning techniques [8].

A more recent method is to combine fuzzy representation, and in particular its ability to provide comprehensible descriptive language, and its approximate reasoning techniques, with decision trees. One of such combinations is implemented in FID [2]. That system is capable of processing a mixture of symbolic, numeric, and fuzzy-termed data. However, all domains must be partitioned into fuzzy sets - by the user. While, in many applications, domain partitioning is determined by external considerations, in some applications this requirement placed an undue burden on the user. Thus, FID3.0 proposed, among other improvements, a capability to automatically construct fuzzy partitions. This paper describes this mechanism, and it provides a comparative experiment.

2. Decision Trees in Supervised Learning

In supervised learning, a sample is represented by a set of conjunctive features expressed in some descriptive language. The objective is to induce decision procedures with discriminative bias - for classification of other samples. Following the comprehensibility principle, which calls upon the decision procedures to use language and mechanisms suitable for human interpretation and understanding [5], symbolic systems remain extremely important in many applications and environments. Among such, decision trees are one of the most popular.

Decision tree methods use recursive partitioning procedures to build decision trees. Subsequently, they use matching inference procedures for classification of new samples.

ID3 [6], and its successor C4.5 [8], along with CART [1], are the two most widely used decision trees. Their basic ideas are the same: partition the sample space in a data-driven manner, and represent the partition as a tree. Examples of the resulting trees are presented in Figure 1, built from examples of two classes, illustrated as black and white. An important property of these algorithms is that they implicitly attempt to minimize the size of the tree while optimizing some local quality measure - such as entropy or information contents [1][6][8].

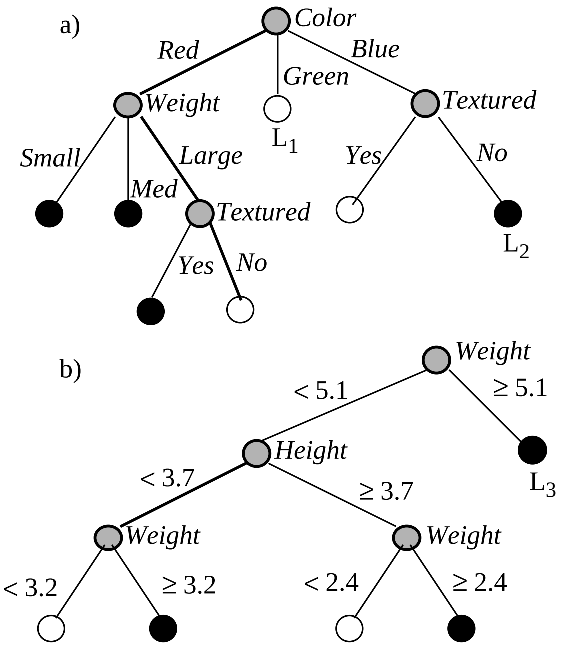
The tree is built in a data-driven algorithm. This algorithm is usually depth-first, meaning that the direction of building (the choice of node to be expanded) has no impact on the final tree. If any changes are to be made (such as pruning), they are performed on the final tree. Tests to be performed on data (corresponding to selecting attributes to be tested in tree nodes) are decided based on some criteria. The most commonly used is information gain, which is computationally simple and shown effective: select an attribute for testing (or a new threshold on a continuous domain) such that the information difference between that contained in a given node and in its children nodes (resulting from splitting according to those tests) is maximized. The information contents is measured according to [6]: $I^N = -\sum_{i=1}^{|C|}(p_i \cdot \log p_i)$, where C is the set of decisions, and p_i is the probability that a training sample in the node represents class i.

1. The root of the decision tree contains all training examples. It represents the whole description space since no restrictions are imposed yet.
2. Work with any node N. The node becomes a leaf when either its samples come from a unique class, when all attributes are used on the path leading to the node, or when possibly information in the node becomes too unreliable (e.g., when too few examples are found). Proceed only when decided to further split the node.
3. Compute the information content at the node N. Then, for each attribute a_i not appearing on the path to N and for each of its domain values a_{ij}, compute the information contents in children nodes restricted by the additional condition $a_i = a_{ij}$. Subsequently, compute a combined weighted information contents in all the children, and the resulting gain with respect to N. Note that for the not-pre-partitioned domains, the algorithm tries all applicable thresholds, selecting the best one - this results in a binary test, as illustrated in Figure 1b. Note that such attributes can appear more than once on a given path (with different binary test).
4. Select the attribute maximizing the gain, and subsequently split node N using the applicable tests.

Many modifications and enhancements of the basic algorithm have been proposed and studied. The most important are other attribute selection criteria, gain ratio accommodating domain sizes [8] (this reduces the utility of domains with large number of domain values to be used for tests, which domains increase discernibility but also reduce comprehensibility), other stopping criteria including statistical tests of attribute relevance, and tree pruning aimed at additional generalization [8].

Afterwards, decision trees use the same basic inference mechanism for classifying new samples. Features of a sample are matched against the tests of the tree, starting from the root and descanting along the matching path. The sample is classified according to the classification of the leaf that it reaches. For example, a new sample with *Color=Green* would be classified according to the classification of L_1 in Figure 1a, and a new sample with any *Weight*$\geq$5.1 would be classified according to the classification of L_3 in Figure 1b.

Figure 1 Examples of decision trees generated by ID3(a) and CART (b). CART builds the tree by recursively selecting thresholds. This technique can be mixed with that of (a) when a mixture of both kinds of domains is present.



3. Fuzzy Decision Trees

FID trees differ from traditional decision trees in two respects: they use splitting criteria based on fuzzy restrictions, and their inference procedures are different. Fuzzy sets defining the fuzzy terms used for building a tree are imposed on the algorithm (after the preprocessing described in the next section).

The FID algorithm can handle data described by various attributes and domains. For example, the same data can be described with nominal attributes, even those inherently symbolic, attributes with continuous domains with no predefined fuzzy sets, such as *Income*, and continuous attributes with user-predefined symbolic terms (or fuzzy terms), such as *Weight* (with potential fuzzy terms: *Light*, *Medium*, *Heavy*). The same mixture can be used as classes. Each piece of data is also augmented with a “confidence” weight (disregarded here in subsequent presentation).

As an illustration, consider attributes *Income*, *Sex*, *Employment*, and class *Credit*. Assume that *Income* does not have a predefined linguistic domain, *Sex* is a symbolic attribute, *Employment* has both continuous domain and a predefined linguistic domain of fuzzy terms *HomeMaker*, *PT*, *FT*, and *Credit* has a continuous domain on (0,1) as well as a predefined linguistic domain of fuzzy terms *Yes* and *No*. Given that, the following are potential data samples:

- *Income* is $55k and *Sex* is *Male* and *Employment* is 25 hours/week -> *Credit* is 0.9
- *Income* is $20k and *Sex* is *Female* and *Employment* is *PT* -> *Credit* is *No*.

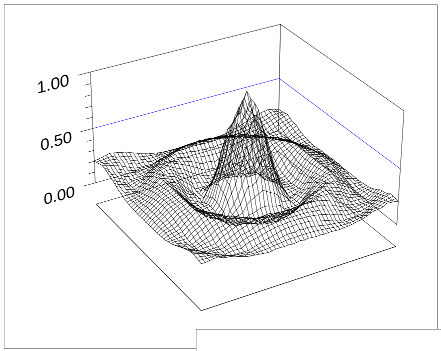
After a preprocessing, as necessary to create fuzzy partitions, FID fuzzy decision tree is constructed similarly to the standard decision tree, with a recursive depth-first procedure. However, there are a number of subtle differences:

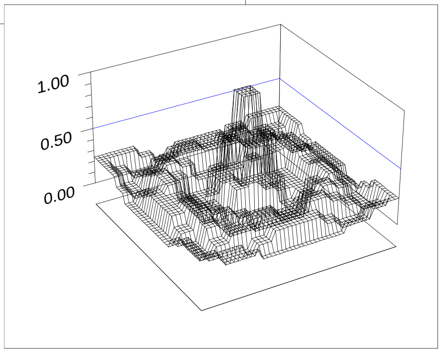
1. Data samples may match more than one test of a node. When aggregated over multiple levels, this leads to samples falling into many nodes, with a real-valued degree (based on the aggregated match to the fuzzy restrictions on the path).
2. The information contents formula is modified to reflect partial memberships (in addition to allowing absent features). For details, see [2].
3. Fuzzy match is determined based on preselected norms, or by selecting best norms from a predefined set.

However, the most profound differences are in the process of classifying a new sample. These differences arise from the fact that

- FID trees have leaves that are more likely to contain samples of different classes (with different degrees of match),
- the inference procedure is likely to match the new sample against multiple leaves, with varying degrees of match.

Figure 2 Illustrations of the knowledge of two FID trees trained for the mexican sombrero function.





To account for these potential problems, a number of inferences routines have been proposed. Some inferences follow the ideas of approximate reasoning [2], other follow machine learning principles of exemplar learning [3]. Some of these inferences have more global character, some are more local and behave like noise filters [2]. Whatever specific inference is used, the outcome is a value from the domain of the class variable.

FID trees have been shown to be capable of producing knowledge which is both comprehensible yet capable of generating finer levels of detail - depending of the actually used inferences. For more information, see [2].

As an illustration of the descriptive power of FID trees, consider the well known mexican sombrero function [9]. When the FID tree is trained with data samples from a 13x13 grid, suing domains with predefined 13 fuzzy terms, two of its interpretations, following two different inferences, are illustrated in Figure 2.

4. Fuzzy Partitioning

If at least one attribute does not have a predefined fuzzy partitioning (does not have the linguistic domain), data-driven preprocessing is invoked in order to attempt to partition such domains for the relevant attributes (and not necessarily all such attributes).

All attributes with predefined partitions retain their partitions (there is no optimization here - the assumption is that if the user predefines a partition, there are some external factors determining that). The remaining attributes are partitioned with fuzzy sets. However, there is no guarantee that a specific attribute will have its domain partitioned, or to what degree of detail a domain will be partitioned (how many fuzzy sets will be generated). Only attributes determined relevant to the classification task (determined locally in the context of partitioning the descriptive space) will have their domains partitioned. Of course, the class variable must have a linguistic set (either fuzzy or symbolic).

The partitioning process builds a fuzzy tree, while creating new partitions on those attributes which lack predefined partitioning. Following the same techniques to reduce gains for attributes with too many values, as that currently used in FID [2] and C4.5 [8], pressure is applied against creating too many partitions over a single domain (to keep comprehensibility under control). Except for this pressure, and for its queue-based procedure, this process is similar to that of finding thresholds, as currently implemented in C4.5 - except that the actual thresholds here are not explicit but they result from intersections of the constructed fuzzy sets. Once the tree is built, it is abandoned - only the resulting partitions are carried on to the regular building process (the regular tree-building process follows different procedures, and thus it can result in different trees). The partitioning procedure can be detailed as follow:

1. Start with existing partitions on the pre-partitioned attributes. On the remaining attributes (call them potential), create a single fuzzy set - rectangular normal set spanning the whole domain. This will ensure consistent treatment of all attributes in the attribute-selection process, yet it does not introduce any bias for such attributes since all potential samples will have the same membership in such sets - and thus such sets will never be selected for testing in a node - until they are split into more fuzzy sets.
2. Start in the root node, put it on a priority queue.
   A node contains the training samples matching the fuzzy restrictions leading to the node (with membership values according to those aggregated matches).
   The queue is a priority queue, ordered by the amount of data contained in the nodes (more data first) - to ensure that domain partitioning is carried out with the most relevant information.
3. Select the next node from the queue, or stop when no more available.
4. Perform the usual process of attribute selection for splitting the node, except that in addition to evaluating all current attributes (including pre-defined and potential attributes), with their current linguistic domains, the procedure also attempts to split any fuzzy set over the domain of a potential attribute, in the context determined by the path to the node.
   The context (path) determines the subdomain - by means of the fuzzy restrictions leading to the node. For example, if the fuzzy restriction *Weight* is v_1 (v_1 is new fuzzy term for the potential attribute *Weight*) is leading to the current node, then only the subdomain for the base of the fuzzy set v_1 is the potential place where the fuzzy set for v_1 can be split.
   Attributes with predefined partitions may only appear once on a path (thus some such attributes can be excluded from consideration in the given node). Potential attributes are always tried.
   Potential attributes with a new fuzzy set have their gains reduced to account for the increase in their linguistic domain size.
5. Select the best attribute (and potentially the best new fuzzy partitioning if the best is a potential attribute) with respect to gain. Split the node on this attribute, remove the node from the queue, and insert the newly generated children nodes according to the priority criterion. However, do not insert nodes whose information contents is such that they cannot potentially bring much gain (information measure is close to 0). This ensures that we do not create new fuzzy sets based on unnecessary tests.
   Record the new fuzzy set for the attribute.
6. Repeat from step 3.

Afterwards, perform the usual process of building the fuzzy tree (starting all over).

5. Illustrative Experiment

For illustration, we have selected three widely used standard data sets from the machine learning depository at Irvine [4]: Iris, Bupa, and Pima. For all cases, we assumed that continuous domains are not pre-partitioned, letting FID3.1 generate fuzzy partitioning and then build the fuzzy trees. All experiments were performed with 10-fold cross-validation, while measuring comprehensibility (defined as the number of nodes) and predictive accuracy on unseen data. To compare results, we used C4.5 in exactly the same settings.

Table 1 Resulting predictive accuracy on unseen data.

Algorithm	Iris	Bupa	Pima
C4.5	94.0%	67.9%	74.7%
FID3.1	96.0%	70.2%	75.9%

The results are presented in Table 1 and Table 2. It is easy to observe that FID3.1 produced consistently higher results, both in terms of tree size and in terms of predictive accuracy on unseen data.

Table 2 Resulting comprehensibility (average number of tree nodes).

Algorithm	Iris	Bupa	Pima
C4.5	5.0	34.4	45.2
FID3.1	5.0	28.9	23.1

6. Summary

We have presented a method to partition continuous or large-valued domains into fuzzy sets. The method is data driven and each domain is partitioned in a data-driven manner, in the context of learning decision trees - only partitions determined relevant to the task are ever attempted. The method proves itself on a number of comparative experiments against the well known C4.5 decision tree. Future extensions should include tree pruning and rule generation capabilities.

A public release of `FID3.1` is planned very shortly at `http://www.cs.umsl.edu/~janikow/fid/`.

References

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

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

[3] C.Z. Janikow. Exemplar Learning in Fuzzy Decision Trees. *Proceedings of FUZZ-IEEE 1996*, pp. 1500-1505. Invited by Prof. Bouchon-Meunier.

[4] C.J. Merz, P.M. Murphy. Repository of machine learning databases. Univ. of CA, Dept. of Information and Computer Science, 1996.

[5] 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.

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

[7] J.R. Quinlan. Unknown Attribute-Values in Induction. In *Proceedings of the Sixth International Workshop on Machine Learning*, 1989, pp. 164-168.

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

[9] I. Suh, Hong & T.W. Kim. Fuzzy Membership Function Based Neural Networks with Applications to the Visual Servoing of Robot Manipulators. I*EEE Transactions on Fuzzy Systems*, Vol. 2, No. 3, 8/1994, pp. 203-220.