Fuzzy Condition Monitoring of Recirculation Fans and Filters

Mike Gerdes
Hamburg University of Applied Sciences
Aero - Aircraft Design and Systems Group
Berliner Tor 9, 20099 Hamburg, Germany
Email: mikegerdes79@gmail.com

Diego Galar
Luleå University of Technology
Division of Operation and Maintenance Engineering
SE-971 87 Luleå, Sweden
Email: diego.galar@ltu.se

University of Skövde
Reliability and Maintenance EngineeringSE-54128, Skövde, Sweden
Email: diego.galar@his.se

Abstract

A reliable condition monitoring is needed to be able to predict faults. Pattern recognition technologies are often used for finding patterns in complex systems. Condition monitoring can also benefit from pattern recognition. Many pattern recognition technologies however only output the classification of the data sample but do not output any information about classes that are also very similar to the input vector. This paper presents a concept for pattern recognition that outputs similarity values for decision trees. Experiments confirmed that the method works and showed good classification results. Different fuzzy functions were evaluated to show how the method can be adapted to different problems. The concept can be used on top of any normal decision tree algorithms and is independent of the learning algorithm. The goal is to have the probabilities of a sample belonging to each class. Performed experiments showed that the concept is reliable and it also works with decision tree forests (which is shown during this paper) to increase the classification accuracy. Overall the presented concept has the same classification accuracy than a normal decision tree but it offers the user more information about how certain the classification is.

Keywords: Fuzzy Decision Trees, Post-Fuzzyfication, Condition Monitoring, Aircraft

2. Introduction

Systems can be complex and difficult to monitor. A good condition monitoring does not only depend on good sensors and a good model but also interpreting of the data. Interpreting sensor data however is not a trivial task. Classification and condition monitoring as shown in this paper also reduces the amount of information that needs to be monitored. Often an expert is needed to interpret the data and make a meaningful "classification". Another problem with a “crisp” classification is that the user gets no knowledge about the “stability” of the classification. Stability in this case mean how fast the classification can change when the input data changes. This however is important information when working with sensors, because it might be possible that small sensor errors can cause a misclassification.
Another problem that motivated the research in this paper is from failure diagnosis. If the classes represent a failure then it is useful for failure diagnosis to know which classes are similar to the current class, because it might be that these failure classes are also responsible for the current visible failure effects.

Basically classification maps an input vector to a class based on a learned or given pattern. In case of system monitoring the class can be a failure or condition of a system.

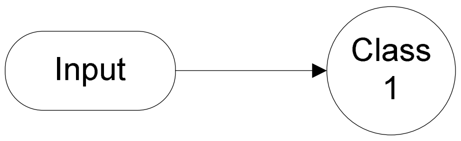

Figure 1: Common classification mapping of one input vector to one class

Most classifiers map an input vector to one output class (Figure 1; the input vector is mapped to "class 1"). However, for monitoring systems and reducing NFF (No Failure Found) failures knowing the probability of an input vector belonging to all possible classes, instead of only the most likely class (Figure 2) is useful. The input vector is still mapped to "class 1", but the input vector also matches the pattern of "class 4" in 89% of the criteria for "class 4". Artificial Neural Networks (ANN) can output the similarity of an input vector to other classes, if one output node is available for every class. But ANNs have their own disadvantages compared to other methods. There are several different methods for calculating the similarity of signals. Many of these methods are used in speech recognition [1].

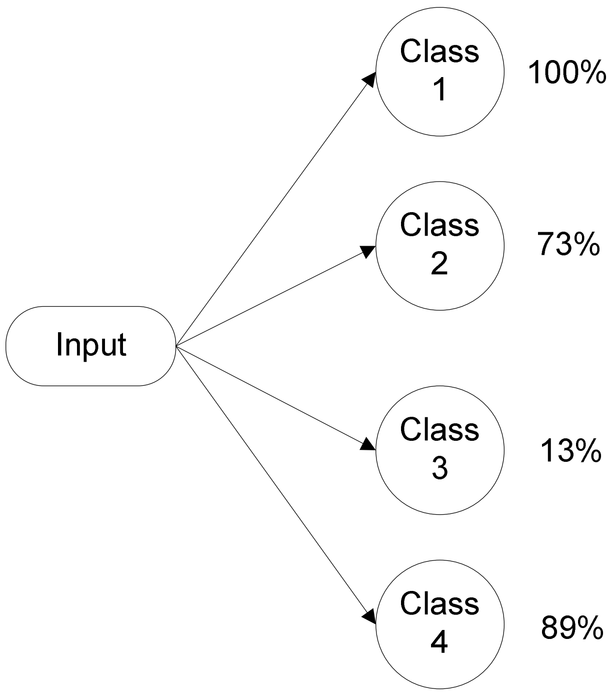

Figure 2: Classification mapping of one input vector to one class and output of similarity

Decision trees are simple and fast classifiers with feature extraction, learning and a high robustness. Decision trees are a method from the area of artificial intelligence and are used for decision making and classification. They are often binary trees, where each node has an if-then-else function on an attribute of the sample data. Advantage of decision trees is the simple structure, the fast calculation and the inherent feature extraction. The ID3 algorithm (Iterative Dichotomiser 3, published by J. Ross Quinlan in 1986, used to generate decision trees [2]) was the first algorithm to construct decision trees. ID3 had some problems and was improved. The improved version of ID3 is C4.5 [3]. It enhances the ID3 algorithm with the ability to handle both discrete and continues attributes, it can handle samples with missing attributes and supports pruning of the tree at the end of the algorithm (removing branches from the tree). The algorithm to build a decision tree uses the concept of information gain to choose attributes from the data and build the tree. Output of a decision tree is only the most likely class for one data sample.

To get more information out of the classification the decision tree inference algorithm was modified to output the probabilities of all trained classes. This modification was done without changing the learning algorithm for decision trees and can be used with any binary decision tree.

Carrying these ideas further, this paper shows how a fuzzy inference of decision trees using numerical attributes can be used to gain more information about a system than just the current condition.

3. Decision Trees

Decision trees are a method from the area of artificial intelligence and are used for machine learning. They are often binary trees, where each node has an if-then-else function on an attribute of the sample data. The ID3 algorithm (Iterative Dichotomiser 3, published by J. Ross Quinlan in 1986, used to generate decision trees [2]) was the first algorithm to construct decision trees. ID3 had some problems and was improved. The improved version of ID3 is C4.5 [3]. It enhances the ID3 algorithm with the ability to handle both discrete and continues attributes, it can handle samples with missing attributes and supports pruning of the tree at the end of the algorithm (removing branches from the tree).

Decision trees are in the proposed method used to calculate and order the features based on the information gain of each feature. During the method validation they are used for failure classification to show the influence of different features on the classification performance.

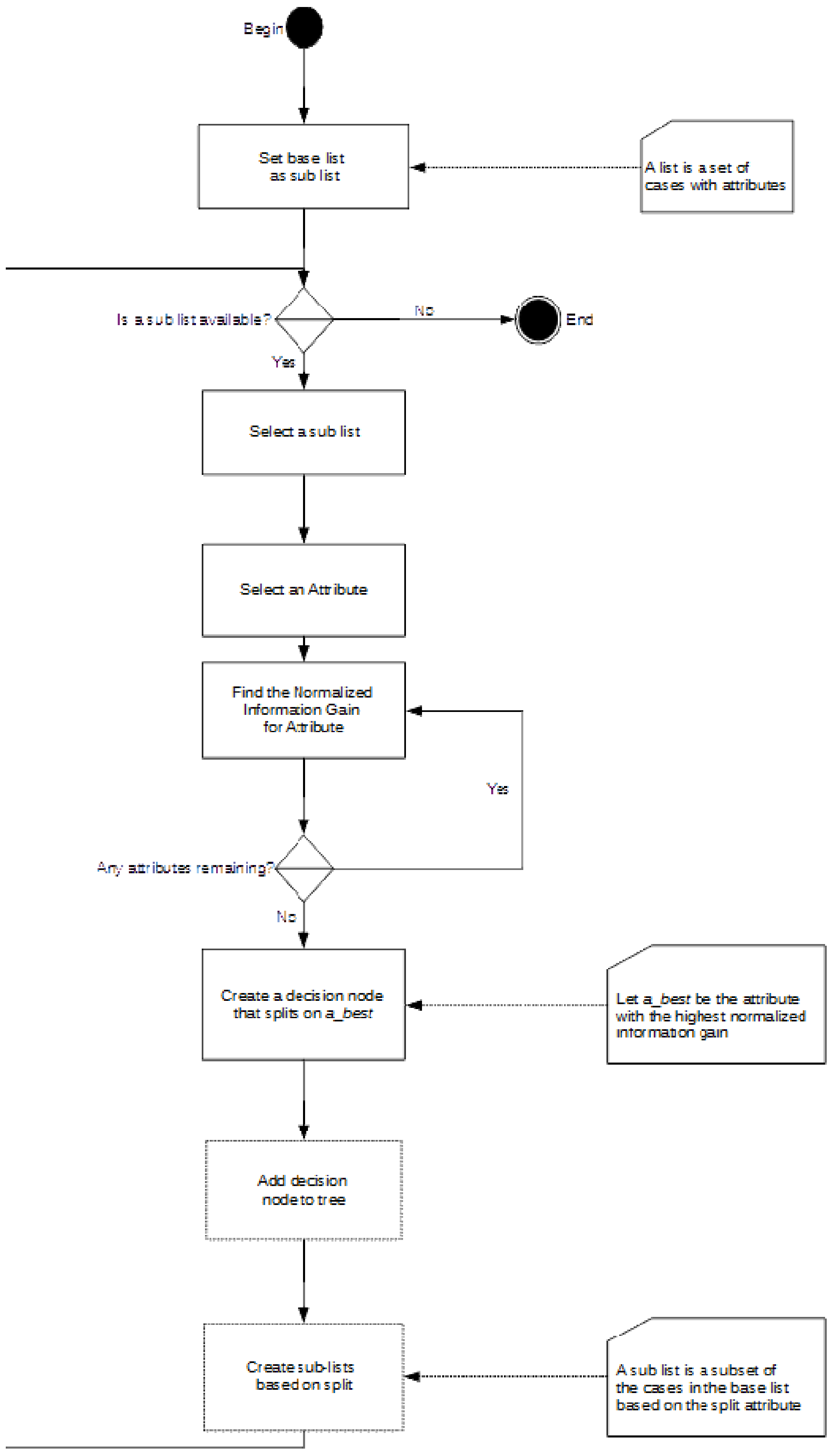
Begin
Set base list as sub list
A list is a set of cases with attributes
Is a sub list available?
No
End
Yes
Select a sub list
Select an Attribute
Find the Normalized Information Gain for Attribute
Yes
Any attributes remaining?
No
Create a decision node that splits on a_best
Let a_best be the attribute with the highest normalized information gain
Add decision node to tree
Create sub-lists based on split
A sub list is a subset of the cases in the base list based on the split attribute

Figure 3: Decision Tree Algorithm Flow Chart

The result of the algorithm is a binary decision tree, where the root of the tree is the attribute with the highest normalized information gain. Nodes in the following levels of the tree represent attributes with lower normalized information gain. If pure information gain is used for splitting, then classes with the most cases are favoured [3].

Information entropy is the knowledge that is contained in an answer depending on one's prior knowledge. The less is known, the more information is provided. In information theory information entropy is measured in bits. One bit of information entropy is enough to answer a yes/no question about which one has no data [4]. The information entropy is also called information and is calculated as shown below. $P(v_i)$ is the probability of the answer v_i.

$$I\left(P(v_i), \dots, P(v_n)\right) = \sum_{i=1}^{n} - P(v_i) log_2 P(v_i) \qquad (1)$$

The information gain from an attribute test is the difference between the total information entropy requirement (the amount of information entropy that was needed before the test) and the new information entropy requirement. p is the number of positive answers and n is the number of negative answers [4].

$$Gain\ (X) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} \times I\left(\frac{p_i}{p_i + n_i}, \frac{n_i}{p_i + n_i}\right) \qquad (2)$$

C4.5 uses the normalized information gain or the gain ratio. Split info is the information that is gained from choosing the attribute to split the samples.

$$Split\ Info\ (X) = -\sum_{i\,=\,1}^{n} \frac{p_i + n_i}{p+n}\ log_2 \left(\frac{p_i + n_i}{p+n}\right) \qquad (3)$$

Gain ratio is the normalized information gain and is defined as shown in equation (4 [3].

$$Gain\ Ratio\ (X) = \frac{Gain\ (X)}{Split\ Info\ (X)} \qquad (4)$$

Pruning is the reduction of the depth of a decision tree. The tree gets better at classifying unknown samples, but might get worse at classifying the test samples. Normally pruning increases the overall classification accuracy, but too much pruning can increase the number of false classifications.

Decision trees are good for diagnostics in the context of condition monitoring. They classify data with low computation needs and the generated decision trees are highly comprehensible by humans. Another advantage of decision trees for condition monitoring is that they can be transformed into simple logical equations for each class that can be checked and modified by a human expert.

Decision trees are used to solve a large variety of problem e.g. tag speech parts [5], land cover mapping [6], text mining [7] or condition monitoring [8] [9] [10].

3.1.Fuzzy Decision Trees

Decision trees can also be evaluated and created using fuzzy rules and concepts. Most often fuzzy attributes and values are used to create a fuzzy decision tree that operates on fuzzy sets. Fuzzy decision trees can be used to overcome some limitations of decision trees (where some of the available features are real- or multivalued, or a numerical decision is needed [11] and that small value changes can change the classification result [12]. Wang, Zhai and Lu [13] use fuzzy decision trees for database classification using rough sets. Yuan and Shaw suggest the following fuzzy decision tree induction [14]:

1. **Fuzzifying the training data**. The data in the data set is converted into a fuzzy set using a member ship function. Salary data, for example, can be converted into three groups, low, average high. Each salary would have a value between 0 and 1 that defines how good a class represents this data. Membership functions can come from mathematical, expert or statistical sources. [14]

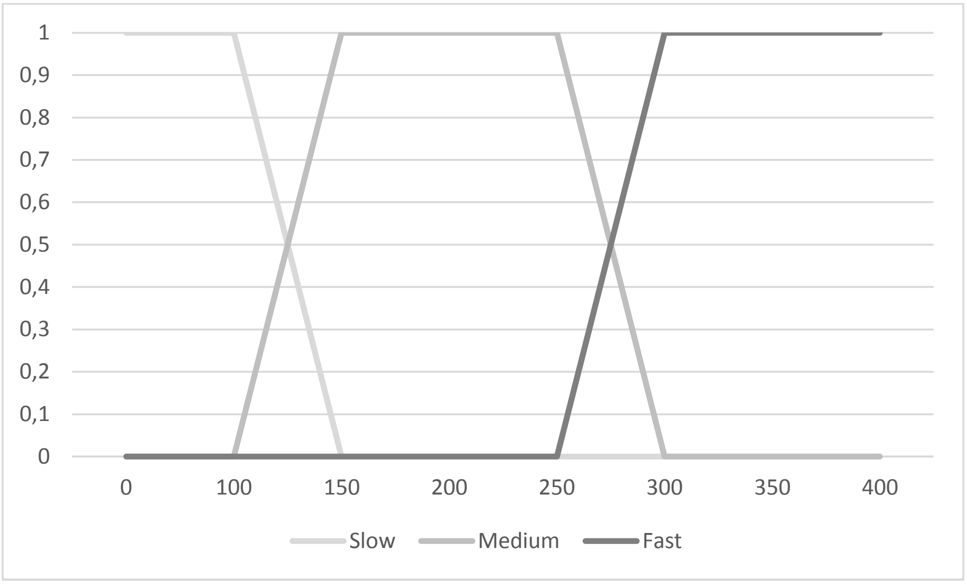

Figure 4: Sample Fuzzy Member Functions for Speed

2. **Inducing a fuzzy decision tree.**
„Step 1: Measure the classification ambiguity associated with each attribute and select the attribute with the smallest classification ambiguity as the root decision node.
Step 2: Delete all empty branches of the decision node. For each nonempty branch of the decision node, calculate the truth level of classifying all objects within the branch into each class. If the truth level of classifying into one class is above a given threshold/~, terminate the branch as a leaf. Otherwise, investigate if an additional attribute will further partition the branch (i.e. generate more than one nonempty branch) and further reduce the classification ambiguity. If yes, select the attribute with smallest classification ambiguity as a new decision node from the branch. If not, terminate this branch as a leaf. At the leaf, all objects will be labelled to one class with the highest truth level.
Step 3: Repeat step 2 for all newly generated decision nodes until no further growth is possible, the decision tree then is complete.“ [14]

3. **Converting the decision tree into a set of rules**. Fuzzy decision trees can be converted into logical rules like crisp decision trees. Each path of the tree is converted into a single rule that represents the attribute decisions at each passed node until a leaf is reached.
4. **Applying fuzzy rules for classification**. Only one path/rule is evaluated if a crisp decision tree is evaluated. For a fuzzy decision tree evaluation each path is evaluated and for each path is a fuzzy result calculated using the rules and fuzzy rules.

Janikow [11] shows a slightly different method for fuzzy decision tree induction and explains how a fuzzy decision tree can be optimized so that the classification accuracy is improved.

The previously presented method falls into the category of pre-fuzzyfication, where the data is fuzzyfied before the decision tree induction [15]. Post-fuzzyfication is the fuzzyfication of the generated decision tree rules.

3.2.Concept

The presented concept uses post-fuzzyfication to change the sample classification of and existing decision tree. One result per possible class is returned instead of one single classification result. The multiple results represent the similarity of the sample to each class. A decision tree is generated traditionally, but uses a fuzzy inference for the inference of an input vector:

1. Generation of feature vectors
2. A decision tree is generated with C4.5 based on labelled data samples.
3. The decision tree is evaluated using the fuzzy inference concept.

The fuzzy inference calculates an output for every leaf of the tree. Every output value of a leaf is a value between 0 and 1 and represents the similarity of a sample to the class that is associated with the leaf (see **Error! Reference source not found.** for an example). If multiple leafs are associated with the same class, then the leaf with the higher value is taken. The value that is return is the similarity of the sample to the class. Similarity is a function of the distance of an input vector to a class. The similarity is calculated based on a weighting function of the decisions taken (node inferences) to classify the sample. For condition monitoring and maintenance the similarity can indicate possible other faults and conditions of a system. The class with the maximum similarity of 1 is still the same class that the C4.5 algorithm would generate. The proposed fuzzy inference works as follows:

1. Every path between two nodes or a node and a leaf has two labels: *PathDepth* and *PathWeight*.
2. Start at the root node
3. *PathDepth* and *PathWeight* are 0 for the root node
4. Evaluate the node condition
5. Calculate path labels:
    a) If the test of the condition is true then label the *True* path with *PathDepth* +1 and *PathWeight* + 1.
    b) If the test of the condition is false then label the *False* path with *PathDepth* +1 and *PathWeight* + 1.
    c) Label the other path to child-nodes with *PathDepth* +1 and *PathWeight* + $nodeweight(AV, SV)$. See Equation **Error! Reference source not found.** and (
6. Choose a new node, with a labelled path to its parent.
7. Use the path labels for *PathDepth* and *PathWeight*.
8. If the node is no leaf then continue at step 4.

9. If the node is a leaf then return $\frac{PathWeigth}{PathDepth}$ and the leaf label and continue with another node.
10. If multiple leafs return values for a class then take the higher value.

The path from one node to another is labelled with the taken decisions. Weights are based on the distance of the attribute value from the sample value in the observation. The highest calculated value (between 0 and 1) for every possible decision is returned at the end of the inference. Thus all possible paths are evaluated and we gain a measure of similarity of an input vector to all classes. Advantage of this approach is that the similarity of a data sample to different conditions can be calculated and that the decision tree generation algorithm does not need to be changed. It should be noted that the concept is designed in such a way that the characteristics of one attribute (mean, minimum, maximum …) does not need to be known. In addition, the input vector for training and classification does not have to be modified in any way to fit the new algorithm.

It is assumed that the tree is a binary tree with numerical attribute values. Returned values of leafs are between 0 and 1. The weight for each node (*nodeweight*) is calculated as shown in Equation 5. Equation (limits the function values. *nodeweight* can be between 0 and 1. *AV* (attribute value) is the value of the sample for the condition of the current node. *SV* (split value) is the value of the node, which marks the border of the condition e.g. the split value of "$x \leq 17$" is 17. Figure 5 shows the nodeweight, if the decision is "false" otherwise the nodeweight is 1.

$$nodeweigth(AV,SV) = \begin{cases} 0 & if\ nodeweigth(AV,SV) < 0 \\ 1 - \frac{|AV-SV|}{2SV} & if\ 0 \leq\ nodeweigth(AV,SV) \leq 1 \\ 1 & if\ nodeweigth(AV,SV) > 1 \end{cases} \quad (5)$$

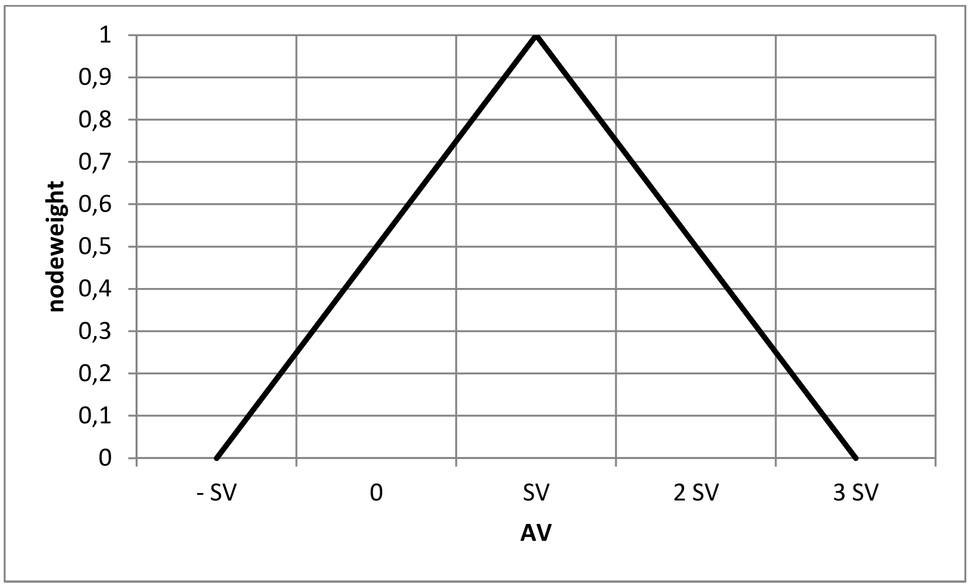

Figure 5: nodeweight for "false" decision

It is possible to use a different weighting function. The requirement is that the weight for each node needs to be between 0 and 1 for the algorithm. If the weight is higher than 1 the result at a leaf can be higher than 1. The given weighting function was chosen for different reasons. One reason was that a black box approach for used for the monitored system and it is unknown what input values of vectors mean. It cannot be assumed that the training vectors for the algorithm contain the max or min values or even represent an average. These circumstances make it difficult to use absolute values. The only values that are available without adding additional information or calculation to the algorithm is the split value of a node. Also the correct classified class needs to have a value of "1" at the corresponding leaf. Choosing $2SV$ as the limit where the weight will be 0 was an arbitrary choice which is based on some tests with input vectors. It is possible to use a higher or lower value. If the maximum and minimum values for features are available, then it is possible to use those as limits and use Equation 2.

$$nodeweigth(AV,SV) = \begin{cases} 1 - \frac{SV - AV}{SV - min} \; if \; AV \leq SV \\ 1 - \frac{AV - SV}{max - SV} \; if \; AV > SV \end{cases} \tag{6}$$

A disadvantage Equation 6 is that every decision returns a *PathWeight* of more than 0, which can result in quite high similarity values. The concept was designed with only numeric values in mind. However, it is possible to use Boolean and discrete values as well with a small modification of the process.

Nodes with Boolean attributes: If Boolean attributes are used instead of numerical values then the weight is assumed to be 0.

Nodes with discrete values: If a node does have more than two children (one for every possible value of the attribute) then only the path linked to the test of the node condition is weighted with 1. For every other unlabelled path to a child the weight needs to be calculated separately.

3.3.Fuzzy Decision Tree Inference Example

An example of the process is shown in Figure 6. Each path from one node to another is labelled with *PathWeight* and *PathDepth* in the form of $\frac{PathWeigth}{PathDepth}$ e.g. $\frac{2}{5}$.

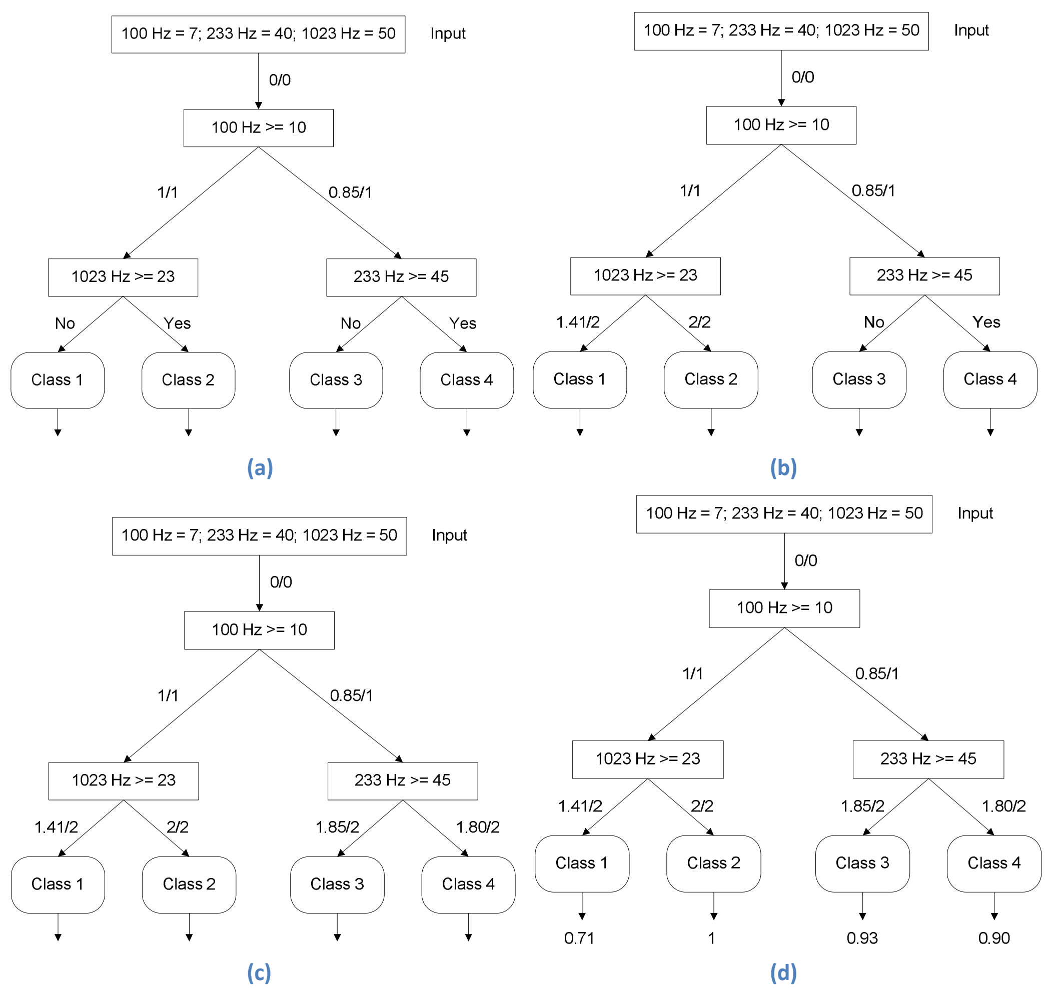

Figure 6: Fuzzy decision tree Inference example

The input vector (values of the power spectrum of the transformed input signal) contains the power (energy per unit time) of the frequencies at 100Hz, 233Hz and 1023Hz. At the first node the 100Hz value is checked if it is larger or equal than 10. The power is not larger than 10 (it is 7) so right path which is the *False* path get +1/+1. For the other path a 1 is added to the *PathDepth* and a 0.85 (the similarity) to the *PathWeight (*Figure 6 (a)). In the next step the 1023Hz node is evaluated. The input vector does have a power at 1023Hz which is higher than 23, so the *True* path gets +1/+1 for a total of 2/2 (+1/+1 to the 1/1 from the parent path). The other path gets a +0.41/+1 for a total of 1.41/2 (Figure 6 (b)). The same process is done for the right hand node (233Hz). Evaluating the node gives a +1/+1 to the *False* path and a +0.95/+1 to the other path (Figure 6 (c)). In the last step the weight of the leafs and the classes are calculated. The input vector is classified as class 2. The similarity to class 1 is 0.71, 0.93 to class 3 and 0.9 to class 4 (Figure 6 (d)). A similarity of 0.93 means that the values do not have to move much to switch the classification result to class 3.

Only the attributes, which define a certain class, are evaluated during the inference. Attributes, which the algorithm did not include in a decision path, are not checked. E.g. after the root node either 1023Hz or 233Hz is checked for classification of a class. So a class is either defined by 100Hz and 1023Hz or by 100Hz and 233Hz, but not by all three attributes. Which attributes are in a decision path and are checked is decided by the decision tree generation algorithm (in this case C4.5). For more complex examples an attribute can be checked multiple times (with different decision values) in a decision path and attributes, which were neglected earlier, are checked again. However even in the simple example all attributes are at least evaluated once, because the algorithm checks all paths. And the inference results are used to calculate the similarity values.

3.4.Fuzzy Decision Tree Forest Inference

A decision tree forest can also be evaluated using the proposed concept. Each decision tree in the forest is evaluated separately using fuzzy decision tree inference. If all trees are evaluated, then the results (similarities) for a class from all trees are added together and are divided by the number of trees (taking the average of a class over all trees in the forest). This is done for all available classes.

4. Validation

Experiments were performed to ensure the performance of the concept. Goal of the experiments was to evaluate the fuzzy decision tree inference for a single decision tree and for a decision tree forest.

4.1.Setup

This section details the setup of the validation. The validation is splitted into two parts. First a single decision tree with fuzzy inference is analysed and then a decision tree forest is analysed. Both validations use the same data that was generated on a test rig at Airbus.

4.1.1. Data

Data for the experiments was recorded on a test rig. The test rig resembles a part of the air conditioning system of the A340-600. Focus of the test rig is the High Pressure (HP) recirculation fan and filter. 29 different conditions were recorded. The test rig is equipped with two valves, one valve at each end of the two open tubes. Table 1 shows the possible settings for both valves. 20 samples of one second duration were recorded for every condition of the test setup.

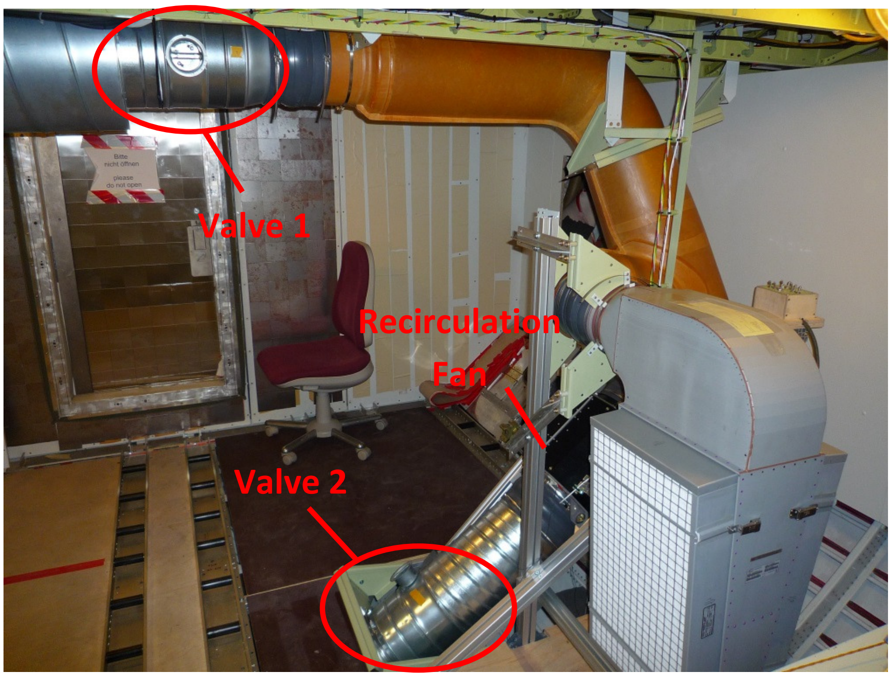

Figure 7: Test Rig at Airbus Operations GmbH

Valve 1	0	15	30	45	60	75	90
Valve 2	0	15	30	45	-	-	-

Table 1: Experiment Conditions

Valve1 is the inlet valve and *Valve2* is the outlet valve. 28 conditions were recorded (Table 1). In addition to these 28 conditions a *not running* condition was recorded. In total 580 samples were recorded. The samples are labelled *valve1/valve2* e.g. 15/0 which represents the test case valve 1 is closed by 15 degree and valve 2 is fully open. Valve 2 was only closed up to 45° to prevent damage to the fan and tubes due to overheating and overpressure.
The test rig also had a filter mounted that was connected to the fan inlet. Thus there was always air flowing to the fan inlet even if valve 2 was completely closed. The fan, filter and the orange tube are original aircraft parts. The positions of the parts were the same as they would be in a real aircraft.

4.1.2. Feature Extraction

The Java software Weka was used to perform the experiments. The decision tree is generated using the C4.5 algorithm. The C4.5 algorithm was used, because it is more advanced than the basic ID3 algorithm (accepts both continuous and discrete features, solves over-fitting problem by pruning handles, incomplete data points) and is available as an open source implementation J48. Input for the decision tree generation is a set of features that was extracted from sensor data. The feature extraction was controlled by different parameters. Table 2 shows the parameter list.

Parameter	**Possible Values**	**Default Value**
Block Width	5/50/100/200	100
Noise Reduction Factor	0/1/2/5	1
Maximum Amplitude	Yes/No	Yes
Mean Amplitude	Yes/No	Yes
Maximum Power	Yes/No	Yes
Maximum Frequency	Yes/No	Yes
Mean Power	Yes/No	Yes
Number of Peaks	Yes/No	Yes
Peak Border	1/2/5	2
Global Maximum Amplitude	Yes/No	Yes
Global Mean Amplitude	Yes/No	Yes
Global Maximum Power	Yes/No	Yes
Global Mean Power	Yes/No	Yes
Global Number of Peaks	Yes/No	Yes
Confidence Factor	0.0001/0.001/0.01/0.1/1	0.001

Table 2: Feature Extraction Parameter

Block Width defines how many frequencies are grouped in the frequency domain to form a block for detailed feature extraction.

The **Noise Reduction Factor** defines how much noise will be reduced. The noise reduction in this concept removes all frequencies in the frequency which power is below *noise reduction factor* times *mean power.*

Peak Border controls what frequencies are defined as peaks. Any frequency which power is greater or equal than the *Peak Border* times the *Mean Power* is defined as a peak.

The **confidence factor** controls how much tree pruning is done and is a parameter of the J48 algorithm of the WEKA software [16]. A confidence factor of greater than 0.5 means that no tree pruning is done. The lower the confidence factor is the more pruning is done.

All other parameters are Boolean parameters which control if a given feature is calculated or not. Elementary feature extraction operations can be executed in any order and allow the creation of a set of feature extraction operations the can be different for each problem [17]. This makes elementary extraction operations also good for machine learning. The used operators are fast to compute and can be used for online monitoring.
These parameters were used to generate a feature vector for each data sample. A genetic evolution algorithm was used to optimize the parameters to generate a decision tree with high classification accuracy. The three best decision trees were used to form a decision tree forest [18]. 20 randomly ordered samples of every class were selected for the decision tree generation.

4.1.3. Single Fuzzy Decision Tree Inference

The experiments themselves were simple. A decision tree was calculated using the data samples and the signal analysis parameters. The decision tree was then evaluated with fuzzy decision tree inference. Test case "15/0" was used to compare the results and was the correct class for all data samples. Five different node weighting functions are tested to be able to evaluate the influence of the node weighting function. The average of all 20 fuzzy inferences of test case "15/0" was taken to reduce the influence of noise.

- Equation **Error! Reference source not found.** as the default node weighting function
- Equation 3 as an example of a function where the weights are always close to 1 and 0
- Equation 4 as a "flat" function with many values close to 1
- Equation 2 as a function that always does have a value $0 < x \leq 1$
- Equation 5 as a function with more values that are 0 and which scales with the value range

$$nodeweigth(AV, SV) = 1 - \frac{|AV - SV|}{0.01SV} \tag{7}$$

$$nodeweigth(AV, SV) = 1 - \frac{|AV - SV|}{10SV} \tag{8}$$

$$\begin{aligned} &if\ (AV\ \leq SV) \Rightarrow\ nodeweigth(AV, SV) = 1 - 5\frac{SV - AV}{SV - min} \\ &if\ (AV > SV) \Rightarrow\ nodeweigth(AV, SV) = 1 - 5\frac{AV - SV}{max - SV} \end{aligned} \tag{9}$$

Each equation (set) replaces Equation **Error! Reference source not found.**. For each equation the same parameter set and decision tree was used.

4.1.4. Decision Tree Forest

A decision tree forest with three trees was created for the experiments with a decision tree forest. Each tree does have a classification accuracy of at least 90%. Every tree uses a different signal processing parameter set, which generates different training vectors and test vectors for the inference. During the experiments the classification accuracy compared to a single decision tree with a classification accuracy of 95% was evaluated. The fuzzy evaluated results for the same 20 data samples and neighbouring classes as in the experiments for the single decision tree were also evaluated.

4.2.Results

This section shows the results of the experiments. First the results for a single fuzzy decision tree inference are shown and second the results for a fuzzy decision tree forest inference are shown.

4.2.1. Single Fuzzy Decision Tree Inference

Table 3 shows the averaged results for five fuzzy decision tree inferences with data samples of class "15/0". The numbers show that the correct class is classified. For comparison the similarity of the three neighbouring classes (0/0, 15/15 and 30/0) is shown. The classes are very similar to class "15/0", but the variance in the results depends on the node weighting function. If a node weighting function is "flat" (meaning that the results are often higher than zero) then the similarities are all close to 1 and have little variance (often between 0.9 and 1). On the other hand, if the function is "narrow" (meaning that often results are produced, which are zero) then the variance is higher and similarities can range from 0 to 1.

Class	Eq. **Error! Reference source not found.**	Eq. (	Eq. (	Eq. (	Eq. (
0/0	0.9585	0.5714	0.9917	0.8294	0.6462
15/0	1	1	1	1	1
15/15	0.9872	0.8357	0.9974	0.9209	0.8505
30/0	0.9996	0.9213	0.9999	0.8	0.8

Table 3: Averaged inference results

This effect occurs because the *PathWeight* is often much higher than zero for "flat" functions. The maximum "narrow" function is, if only 1 or 0 would be valid results. Then the similarity is based on the number of the "True" decision. This may be desirable for some applications, but it hides some information that might be useful for condition monitoring. Small variations of an attribute are not represented in a very "narrow" function, they are filtered out. But often the goal of condition monitoring is not only the similarity but also the ability to detect slight movements in the similarity to predict into which "direction" a similarity is moving if one or more values are modified. Finding a fitting node weighting function depends on the problem and the goals of the application.

Table 4 shows a complete similarity matrix. The matrix contains the results of all leafs of the decision tree for Equation 7. In the matrix the full similarity variance of the results is visible. The table shows that the most similar classes do not always have to be neighbours, but instead they can be classes farther away. Two of the neighbours are very similar, which is the desired result. The similarity of the other classes is mixed. The tendency is that the similarity is lesser if a class does have a higher distance from "15/0".

Valve2/Valve1	0	15	30	45	60	75	90
0	0.5714	1	0.9213	0.5	0.65	0.7583	0.655
15	0.672	0.8357	0.6929	0.6667	0.6512	0.7278	0.4944

30	0.75	0.5	0.5	0.6	0.4444	0.5714	0.4286
45	0.65	0.75	0.5712	0.5667	0.25	0.5	0.25

Table 4: Similarity matrix

4.2.2. Fuzzy Decision Tree Forest Inference

This section shows how the results improve, when a decision tree forest is used. The similarity values of the decision tree forest were calculated by taking the average similarity values of the single decision trees.

4.2.3. Classification Accuracy

The accuracy of a classification is defined as the number of correct classifications (CC) in relation to the number of all classifications (TS), where TS is the number of correct classification plus the number of wrong classifications (see Equation 10).

$$Classification\ Accuracy = \frac{CC}{TS} \tag{10}$$

Classification accuracy is a number between 0 and 1, which can be transformed into a percentage value. The accuracy of the decision tree inference is not influenced by the fuzzy decision tree inference. This is because of the fact that the *True* paths are weighted with "1" while all other paths are weighted with a positive number lower than 1 ($0 < x \leq 1$). The *True* path will always have the highest weight. However it is possible that the classification accuracy of fuzzy decision tree inference is slightly lower than for standard decision tree inference because of the limits of numerical computations. If the weight of a node is very close to 1 then it is possible that due to rounding errors it is counted as a 1 instead of a value that is lower than 1. This case happened in some experiments and was more frequent, if a 32 bit Java floating point data type was used instead of 64 bit Java floating point data type. Using a forest with three decision trees increased the classification accuracy from 95% per single tree to 99% for the complete forest. This is a significant classification accuracy improvement, which comes at the cost of three times the calculation time. However in the example application of air filter monitoring the time is not a critical factor.

4.2.4. Fuzzy Decision Tree Inference

Comparing the results of the same four classes that were evaluated for a single decision tree we get Table 5. The results are similar to a single decision tree with fuzzy inference, but the average similarity and the overall classification accuracy is higher. These results show that the fuzzy inference also works with a decision tree forest. But if one value changes in the input changes then the influence on the similarity is less. Decision tree forests with fuzzy inference are better at calculating the overall similarity of an input, because different signal analysis steps are used and different trees are evaluated. Thus different features are checked and used to calculate the similarity result of a single decision tree.

Class	Eq. 5	Eq. 8	Eq. 9	Eq. 7	Eq. 10
0/0	0.9654	0.6905	0.9931	0.8725	0.7206
15/0	1	1	1	1	1
15/15	0.9815	0.8665	0.9963	0.9370	0.8736
30/0	0.9976	0.8528	0.9995	0.9286	0.9095

Table 5: Averaged fuzzy decision tree forest inference

Valve2/Valve1	0	15	30	45	60	75	90
0	0.6905	1	0.8528	0.5972	0.4833	0.5950	0.04718
15	0.6843	0.8665	0.7403	0.5637	0.4929	0.5537	0.4127
30	0.6630	0.5650	0.4061	0.6071	0.4648	0.6071	0.6111
45	0.6667	0.6533	0.4904	0.6389	0.4087	0.4401	0.4556

Table 6: Similarity matrix for the decision tree forest

5. Conclusions & Discussion

The concept for fuzzy inference of decision trees, which is shown in this paper can achieve the desired performance and delivers good results. It is possible to calculate a similarity measurement for classes in a decision tree without changing the algorithm to create the tree. However, the design of the node weighting function is important. Paths should be weighted with a low *PathWeight* to get a meaningful similarity measurement. A "flat" node weight function is more sensitive the changes in the input values. On the downside a "flat" function does have a lower variance in the similarity values. "Narrow" node weighting functions have the opposite effect. Fuzzy classification plus decision trees are a powerful tool for condition monitoring. The fuzzy decision tree inference is easily understood, fast and small, which makes it good for use in environments characterized by high safety requirements. With additional optimization of the weighting equation it is possible to increase the variance of the fuzzification.
Another possible variation is to not limit the fuzzfication to the split value and the “false” path, but also weight a part of “true” path and thus having a fuzzy border between both paths. The “true” path would always have a higher value than the “false” path, but it would only have the value 1, if the attribute value is quite away from the split value [15].

6. References

[1] L. Rabiner and B. H. Juang, Fundamentals of Speech Recognition, Prentice Hall, 1993.

[2] J. R. Quinlan, “Induction of Decision Trees,” *Mach. Learn,* pp. 81-106, 1986.

[3] J. R. Quinlan, C4.5: Programs for Machine Learning, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[4] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Pearson Education, 2003.

[5] H. Schmid, “Probabilistic part-of-speech tagging using decision trees,” in *Proceedings of the international conference on new methods in language processing*, 1994.

[6] M. A. Friedl and C. E. Brodley, “Decision tree classification of land cover from remotely sensed data,” *Remote sensing of environment,* vol. 61, no. 3, pp. 399-409, 1997.

[7] C. Apte, F. Damerau, S. M. Weiss, C. Apte, F. Damerau and S. Weiss, “Text mining with decision trees and decision rules,” in *In Proceedings of the Conference on Automated Learning and Discorery, Workshop 6: Learning from Text and the Web*, 1998.

[8] V. Sugumaran and K. Ramachandran, “Effect of number of features on classification of roller bearing faults using SVM and PSVM,” *Expert Systems with Applications,* vol. 38, no. 4, pp. 4088-4096, 2011.

[9] M. Saimurugan, K. Ramachandran, V. Sugumaran and N. Sakthivel, “Multi component fault diagnosis of rotational mechanical system based on decision tree and support vector machine,” *Expert Systems with Applications,* vol. 38, no. 4, pp. 3819-3826, 2011.

[10] N. Sakthivel, V. Sugumaran and B. B. Nair, “Comparison of decision tree-fuzzy and rough set-fuzzy methods for fault categorization of mono-block centrifugal pump,” *Mechanical systems and signal processing,* vol. 24, no. 6, pp. 1887-1906, 2010.

[11] C. Z. Janikow, “A Genetic Algorithm for Optimizing Fuzzy Decision Trees,” in *Proceedings of the 6th International Conference on Genetic Algorithms*, San Francisco, CA, USA, 1995.

[12] J. Quinlan, “Decision Trees as probabilistic Classifiers,” in *Proceedings of the Fourth International Workshop on MAchine Learning*, P. Langley, Ed., Morgan Kaufmann, 1987, pp. 31-37.

[13] X.-Z. Wang, J.-H. Zhai and S.-X. Lu, “Induction of Multiple Fuzzy Decision Trees Based on Rough Set Technique,” *Inf. Sci.,* vol. 178, no. 16, pp. 3188-3202, Aug. 2008.

[14] Y. Yuan and M. J. Shaw, “Induction of Fuzzy Decision Trees,” *Fuzzy Sets Syst.,* vol. 69, no. 2, pp. 125-139, Jan. 1995.

[15] I.-J. Chiang and J. Y.-j. Hsu, “Fuzzy Classification Trees for Data Analysis,” *Fuzzy Sets Syst.,* vol. 130, no. 1, pp. 87-99, Aug. 2002.

[16] U. o. Waikato, *WEKA,* 2009.

[17] I. Mierswa and K. Morik, “Automatic feature extraction for classifying audio data,” *Machine learning,* vol. 58, no. 2-3, pp. 127-149, 2005.

[18] M. Gerdes and D. Scholz, “Parameter Optimization for Automated Signal Analysis for Condition Monitoring of Aircraft Systems,” in *3nd International Workshop on Aircraft System Technologies, AST 2011 (TUHH, Hamburg, 31. März - 01. April 2011)*, 2011.