Fuzzy decision trees

Catalin Pol

Abstract

Decision trees are arguably one of the most popular choices for learning and reasoning systems, especially when it comes to learning from discrete valued (feature based) examples. Because of the success they acquired in their area, there have been many attempts to generalize the method to better suit working with real-valued, numerical attributes, but also for missing values or even numerical outcomes (i.e. for regression tasks). In this paper, we will present a few methods aiming to combine the increased readability of decision trees with the ability to deal with numeric or inaccurate data provided by fuzzy reasoning.

1 Introduction

In recent years, neural networks have become increasingly popular when it comes to classification problems, due to relative ease of application and abilities to provide gradual responses. However, they lack similar levels of readability[7], which can be a problem, especially if users need to justify and understand their decisions. In these cases, only decision trees managed to get satisfactory results.
Decision trees were popularized by Quinlan[5], along with the ID3 program. Systems based on this approach perform particularly well on symbolic domains. When a numerical outcome is desired, or when numerical values improves the subsequent reasoning, this algorithm is not applicable. One of the most common methods is to pre-partition the values in fixed length intervals (thus, predifining a set of compact intervals that cover the entire data). Subsequent improvements of this algorithm, such as C4.5, are able to deal with numerical data, by choosing a split point based on the learning examples. We will describe how such trees are constructed in more detail Unfortunatelly, this may decrease the accuracy of the classification, especially for values that are close to the limits of the intervals determined during the training step. In order to complement the problems that are posed by the limitations of decision trees, different approaches have been made to work around this issue. One of these approaches is provided by fuzzy logic.
In fuzzy rule-based systems, the symbolic rules provide ease of understanding and 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, being successfully applied in a number of industrial problems[3].
Traditionally, decision trees have two major components: a procedure to build the symbolic tree and a method for decision making. We will present a few

methods to approach both of these components. The rest of this paper is organized as follows: section 2 contains a short introduction to decision trees; a short introduction to fuzzy set theory is covered in section 3, then a method on how to apply fuzzy reasoning is presented in section 4. Finally, section 6 concludes this article with a few thoughts regarding other possible directions in this fuzzy decision tree research field.

2 Decision trees

Decision trees, as their name suggests, are decision support tools that use a tree-like predictive model which map observations about an item on several levels in the tree until reaching the final conclusion regarding the outcome of the desired function. One of the most used algorithms for constructing decision trees has long been the ID3 method[1] introduced by Quinlan[5]. This algorithm tries to construct the smallest classification tree based on a small set of training examples. The main disadvantage is that this algorithm only considers symbolic (discrete) values both for attributes, as well as for the resulting class. Another major disadvantage of this method include the fact that the pre-partitioning requires previous knowledge of the data (how many intervals are required, how to choose the split points between the intervals). Subsequent improvements of the ID3 algorithm, such as C4.5[4] and CART[1] try to deal with intervals in an optimal manner (they always try to split a compact interval into two subintervals, and they accomplish this by choosing the optimal split-point, based of course on the training data).

In this paper, we will discuss a possible extension of ID3 and C4.5 that uses fuzzy logic mechanisms for decisions. Therefore, we will start by presenting the general idea behind these algorithms. The justification behind them lies in the Occam's razor principle: it will always try to construct a node that maximizes the information gain when choosing the attribute for that particular node. However, this does not always produce the smallest tree, and is therefore only a heuristic. The information gain is formalized by the concept of information entropy:

$$E(S) = -\sum_{c \in Cls} f_S(c) \cdot \log_2 f_S(c)$$

Where:

- $E(S)$ is the entropy of the set S
- Cls is the set of all classes encountered in the training set
- $f_S(c)$ is the frequency (proportion) of the instances that have the value c in the set S [2]

Entropy is used to determine which attribute to use at the current node. An entropy of 0 marks a perfectly classified set. A higher entropy means that the

[1] ID3 stands for "Iterative Dichotomiser 3"

[2] by convension, $xlog_2x = 0$

attribute is better suited for a bigger information gain, which is defined by the following formula:

$$G(S, A) = E(S) - \sum_{i=1}^{m} f_S(A_i) \cdot E(S_{A_i})$$

Where:

- $G(S, A)$ is the gain of the set S after a split over the A attribute
- $E(S)$ is the information entropy of the set S
- m is the number of different values of the attribute A in S
- $f_S(A_i)$ is the frequency (proportion) of the items possessing A_i as value for A in S
- A_i is i^{th} possible value of A
- S_{A_i} is a subset of S containing all items where the value of A is A_i

The information gain quantifies the improvement obtained by splitting according to a given attribute. The above formula however, applies only if the attribute for which we are computing the information gain is discrete. If the attribute is numerical, a similar formula is used[3], but, of course taking into account the fact that we cannot operate on all possible values for the given attribute:

$$G(S, A, \alpha) = E(S) - f_S(S_{\leq\alpha}) \cdot E(S_{\leq\alpha}) - f_S(S_{>\alpha}) \cdot E(S_{>\alpha})$$

Here, $S_{\leq\alpha}$ is the subset of S containing the instances with the values for the attribute A less or equal than a so-called "cut-point" α. Similarly, $S_{>\alpha}$ contains the elements for which the corresponding value is greater than α. For choosing the best cut-point, many C4.5 implementations suggest sorting the data samples in S according the attribute under inspection and computing the information gain for all midpoints between consecutive instances. At each step, both the C4.5 and ID3 algorithm choose the attribute that maximizes the information gain. The ID3 algorithm is described below.
Of course, one difference between the ID3 and C4.5 algorithm is that the C4.5 algorithm can reuse a numerical attribute at multiple levels of the tree. For discrete attributes however, one attribute can be used only once on every path from the root down to a leaf. There are of course many other differences, but they are beyond the purpose of this paper. We will describe the details about infering rules in such a tree in section 4.

3 Fuzzy set theory

In order to better understand the notion of fuzzy set theory, we should first take a short review of the classical set theory (also known as crisp set theory). In this case, you can say that an element either belongs to a certain set or it does not. A crisp set $A \subseteq U$ can be defined by a two-valued characteristic function $A_U : U \to \{0, 1\}$, where U is the universe of all possible values of the elements

[3]Between the two algorithms in discussion, only C4.5 can deal with numerical data

Algorithm 1 ID3 (*Examples*, *TargetAttribute*, *Attributes*)

```
create a root node for the tree, Root
A ← the attribute with highest information gain
set decision tree attribute for Root ← A
for all v_i ∈ A do
  add a new tree branch below Root, labeled with the value v_i
  let Examples(v_i) ← the subset of examples that have the value v_i for A
  if Examples(v_i) = ∅ then
    add leaf node for this branch with label = the most common value in the
    examples
  else
    add as subtree for this branch the tree provided by ID3(Examples(v_i),
    TargetAttribute, Attributes − {A})
  end if
end for
```

(sometimes called the universe of discourse). Moreover, the characteristic function of a set uniquely identifies the set. However, in some cases, because of imprecise measurements or even for some lingvistic concepts, such a characteristic function may not suffice. Concepts like 'tall', 'cold' or 'small' are inherently fuzzy concepts and most of the times these are subjective or dependent on the context.

The fuzzy set theory extends the two-valued characteristic function of a set to a real-valued function. So, a fuzzy set A is described by a membership function $\mu_U : U \to [0, 1]$, which represents the degree to which an element $u \in U$ belongs to the set U. Similar to the basic operations of union, intersection, and complement defined in classical set theory[4], such operations are defined for fuzzy sets. Unlike the crisp case, these operators are not uniquely defined. For example, the Zadeh operators for union and intersection are *max* and *min* respectively (corresponding to the most optimistic/most pessimistic of two given membership degrees). On the other land, the Lukasiewicz operators that the sum (bounded by 1) for the union and the product of two membership degrees for the intersection (this approach was consider in order to closer match the operations from probability theory). However, in both these cases, the complement of a given degree x is defined as $1 - x$.

4 Fuzzy decision tree algorithm

A simple way to explain how the rules that are infered from a decision tree is to consider all the leaves of the tree. For each leaf, a conjunction can be easily constructed by reading the labels of the nodes and branches that are encountered starting from the root of the tree down to the corresponding leaf. Then, in order to get the condition for one of the classes, we would normally construct a disjunction between all the leaves that have a value associated with that class. This approach is basically the idea behind the classical decision trees, but we will extend it by using fuzzy logic.

[4]The intersection, union and complement defined in the crisp set theory are sometimes refered to by their equivalent logical operators: AND, OR and NOT, respectively

One of the major disadvantages of classical decision trees is that each tested object will have only one leaf associated with it. Moreover, this leaf is closely related to the input samples. In many cases, it would be a nice feature to have close values somehow related to each other. Also, it would be nice to be able to construct decision trees by using symbolic values (tall/short, hot/cold), but test objects having a numerical value. All these issues are solvable using a fuzzy approach.
Going back to the classical decisions infered by a decision tree, in order to test an object against a given decision tree, we would start from the root of the tree, and go down the tree by testing the attribute corresponding to the current node and following only the branch of the tree that corresponds to the value that has the same value as object we are testing. In fuzzy terms, this would mean to follow only the branch corresponding to the set where our value has a strictly positive membership degree. In the case of crisp decision trees, only one branch will have degree 1 and all the other values will have degree 0, but for our fuzzy approach, we will consider all the branches for which the membership degree of our object to that particular set is non-zero (or, if we are interested in a faster algorithm, for a membership larger than a certain threshold). For each node, we will also have to keep in mind the membership of the object down the current path (from the root of the node, where the membership is always 1, down to the current node). Again, in the case of crisp decision trees, the membership on the current path is always 1 (only one path and only one leaf has membership 1, all the other ones have membership 0 and are therefore ignored). For simplicity, we will reffer to the membership degree of an object to a certain set of attributes as the membership of the object to the corresponding tree node (or to the corresponding path in the tree). This membership can be computed gradually and needs to use an intersection operator for fuzzy memberships. In other words, considering that the attributes A_1, A_2,... A_k were encountered from the root of the tree to a certain node (the node will in this case be on level k), the membership degree of our object (considering that the values for the corresponding attributes are v_1, v_2,... v_k) is:

$$\bigcap_{i=1}^{k} \mu_{A_i}(v_i)$$

If we are testing a symbolic attribute, depending on the semantics of the attribute, we can either test it against numerical values (for example, a temperature of 25^0C might be considered 70% hot and 30% cold), or against symbolic values. Again, depending on the semantics of the attribute, we can either assign a certain degree of similarity between similar attributes or assume they are completely disjoint, as in the case of classical decision trees. If we are testing a numerical attribute, we always have to deal with a condition against a cut point α. In this case, for each such attribute, we can introduce an extra parameter $\beta > 0$. Around each cut point, we say that there is an uncertainty interval of length β. Inside this interval, the condition $x < \alpha$ has a certain membership degree between 0 and 1. If the attribute is missing when testing the object, Quinlan proposed [6] that the best approach is to evenly split an example into all children if the needed feature value is not available. In the end, instead of reaching a single leaf, as in the case of a crisp decision tree, we will reach multiple leaves. The result returned by our tree for our given object can be

computed in various ways, depending on the semantics of the classes. Either way, for each class, we compute the membership of our object. This is easy to compute because for each leaf we have the corresponding value as well as a membership degree associated with it. We just need to compute the disjunction between the leaveas that have a particular value. If the classes are not independent, we can compute a weighted average of the leaf values (the weights being the membership degrees for each leaf):

$$\frac{\sum_{x \in LeafNodes} value(x) * \mu_x}{\sum_{x \in LeafNodes} \mu_x}$$

where:

- $LeafNodes$ is the set of all leafs in the tree (since the membership of non-accessible leafs is zero, they can be excluded from this computation)
- $value(x)$ is the value corresponding to the leaf node x
- μ_x is the membership degree of the given leaf

For example, consider that we have a decision tree that originally classified samples into 3 classes: white, gray or black and the combined leaf memberships 0.45, 0.3 and 0.25 respectively. In some cases, we can consider that a combination of two classes be considered as one of the other classes (in this case, the class "grey" will be the output, even if this is not the most likely class according to our decision tree). In other cases, classes are not comparable, in which case, we will output the class with the highest combined membership among the leaf nodes (in our example, "white" will be the output).

5 Experimental setup

For our testing purposes, we used the J48 classifier provided by the Weka machine learning tool, which is an open-source, freely available implementation of the C4.5 algorithm presented earlier[5]. We chose this in order to generate a decision tree based on a given function. The function we chose for our first test was $sin(x)$, restricted to the interval $[-2, 2]$ (see figure 1). Because we are dealing with only one continuous attribute, the resulting tree was a binary tree with 23 leaves, as seen in figure 2. One of the disadvantages of a decision tree is that the output is not continuous. There are some algorithms, such as CART[1], which produce an estimate for each leaf (based of course on the training samples), but C4.5 has a discrete output. We chose as the output classes 21 nominal strings corresponding to the numbers $\{-1, -0.9, -0.8, \ldots, 0.9, 1\}$. The training data was chosen as the $sin(x)$ function rounded to 1 decimal, on 400 equidistant points in the $[-2, 2]$ interval. The learned function is also depicted in parallel with the original function in figure 1.

As you can see, the function learned is a step function, which is caused by our approximation (keep in mind this tree was constructed by rounding the sin function on 400 points to 1 decimal). Even if such a decision tree might suffice for most applications, we will try to improve it even further. For example, we

[5]Weka is available at `http://www.cs.waikato.ac.nz/ml/weka/`

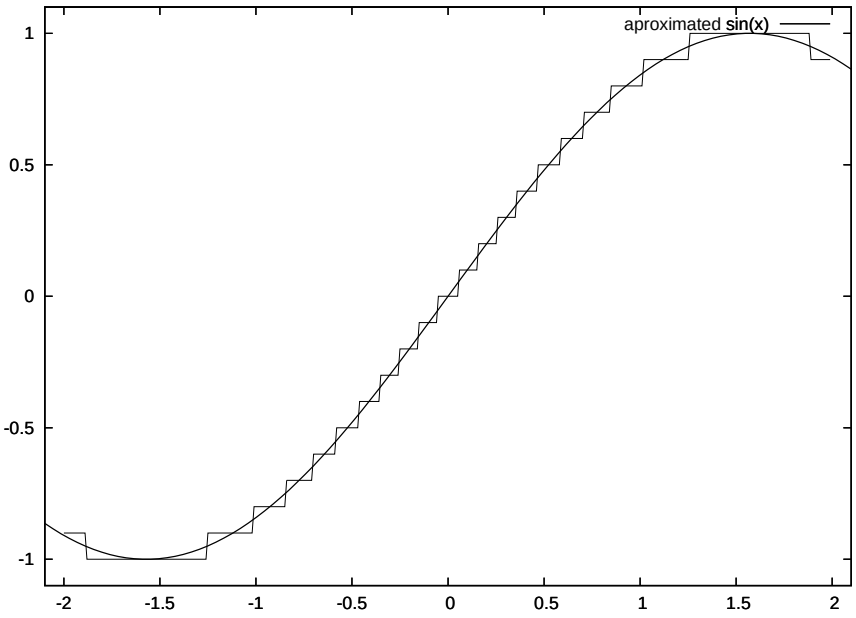

Figure 1: $sin(x)$ as learned by the C4.5 algorithm

tested how well this tree performs on 40000 equidistant points in the same $[-2, 2]$ interval (we tested if the tree correctly predicted the value of the *sin* function rounded to 1 decimal on these numbers). The accuracy on these points is about 97.27%: 1089 instances out of the 40000 were incorrectly classified (however, for all these points, the error was by just 0.1). Furthermore, all of these incorrect instances were near the cut points.
This means that in order to improve the accuracy of our decision tree, we must correct one of the major drawbacks of most supervised learning algorithms. That is, all such algorithms rely on representative, accurate data during the learning phase. In our case, the lack of instances near the actual cut points has lead to incorrectly predicted cut points. The solution we propose is to create a fuzzy cut point. First, let's analyze the characteristic function of a typical cut point:

$$A_{\leq\alpha}(x) = \begin{cases} 1 & \text{if } x \leq \alpha \\ 0 & \text{if } x > \alpha \end{cases} \tag{1}$$

Next, we introduce the notion of a fuzzy cut point, by using an extra positive parameter β:

$$A_{\leq\alpha,\beta}(x) = \begin{cases} 1 & \text{if } x \leq \alpha - \frac{\beta}{2} \\ \frac{1}{2} + \frac{\alpha - x}{\beta} & \text{if } x \in [\alpha - \frac{\beta}{2}, \alpha + \frac{\beta}{2}] \\ 0 & \text{if } x \geq \alpha + \frac{\beta}{2} \end{cases} \tag{2}$$

In other words, near the cut point, there is an "uncertainty" interval of length β, where the characteristic function decreases linearly instead of changing its value from 1 to 0 directly. Since we chose this linear representation in the "uncertain" interval, the value of this function is 0.5 at the original cut point.

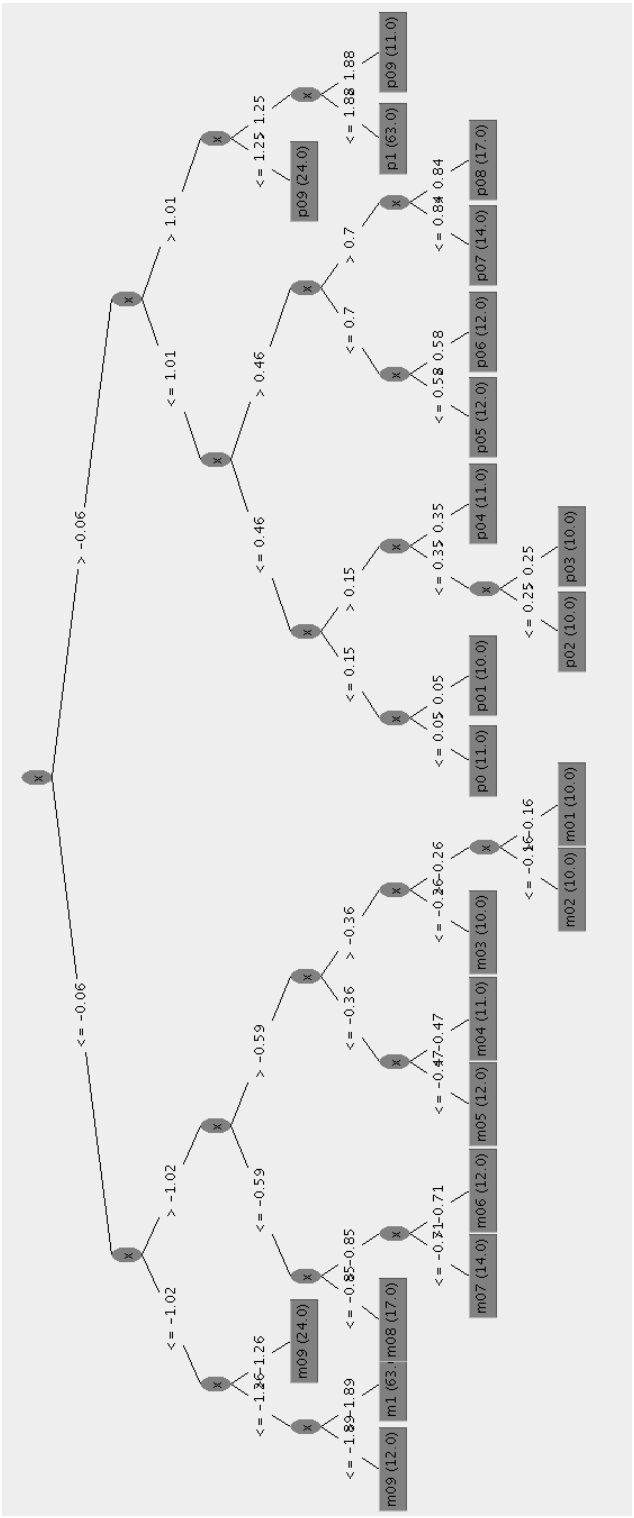

Figure 2: Decision tree generated

Comming back to the decision tree, normally we would construct our rules by starting at the root node and descending in the tree by using the correct branch. In this case, the branch was always unique, because only one branch had the characteristic function equal to 1, all the other ones having the characteristic 0 (if the attribute chosen by the algorithm was numeric, in C4.5, this would correspond to a node with only two branches, but if the attribute was discrete, the node should have one branch for each possible value of the attribute). In our fuzzy cut point model, the "correct" branch is no longer uniquely determined. More exactly, each branch has a truth value associated to it, a number between 0 and 1 (the membership function associated with the condition that labels that branch). As with the classical reasoning in decision trees, we will ignore all branches with a 0 truth value (and their corresponding subtrees), but we will explore all leaves which have a combined membership that is strictly positive. By the combined membership of a leaf node we mean the intersection of the truth values of the labels which are encountered from the root node down to the leaf in question. In our experiment, we used the Lukasiewicz operator for intersection (the product).
The final step is to consider all the non-zero leaves and their membership degrees and to generate the output of the function. One of the easiest solutions for this problem is to take the weighted sum of the leaf values (the weights being the leaf memberships). This also means that the output of our decision tree will no longer be a step function, but a continuous one. In the following pages, we will present a few results obtained using this method (by comparison to the actual $sin(x)$ function), for different values of β (figures 3, 4, 5 and 6). Just as a remainder, the classical decision tree corresponds to a value of $\beta = 0$ (same values as the function in figure 1).

6 Summary and conclusions

This paper proposes a method of interpreting decision trees by using fuzzy logic. Some of the best results achievable with this method are when we are trying to learn a function that is continuous with respect to a given attribute, for which we are considering fuzzy logic. Although a lot of issues are still up for discussion (such as choosing the appropriate fuzzy operators, choosing the crisp partition on each attribute, then the membership function for each attribute). By using this fuzzy interpretation of the nodes, we can achieve greater accuracy when computing the outcome of our classifier. However, this difference between the construction and the interpretation of the resulting tree, can lead to some errors while computing the value for the training data itself. This can sometimes be a big issue, especially around local extremum points, but it can also provide a very valuable piece of information: it can signal where an error in the original data has occured, especially if we are dealing with a regression task. In this case, noisy input data will cause less disturbance in the predicted values, since the general bias of fuzzy decision trees is to towards a "smooth" continuous function. The "smoothness" of the function which is predicted by a fuzzy decision tree comes from the bias of this fuzzy interpretation: similar values in the input should cause similar output. Moreover, while the crisp interpretation of decision trees would normally yield a function with many discontinuity points (usually, the output of such a decision tree is a function which is constant on small intervals,

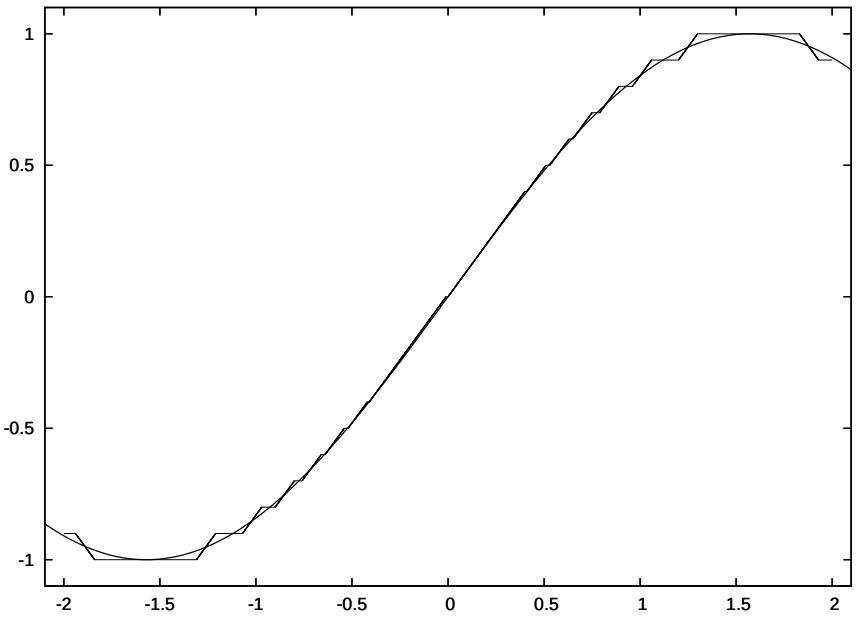

Figure 3: Function obtained for $\beta = 0.1$

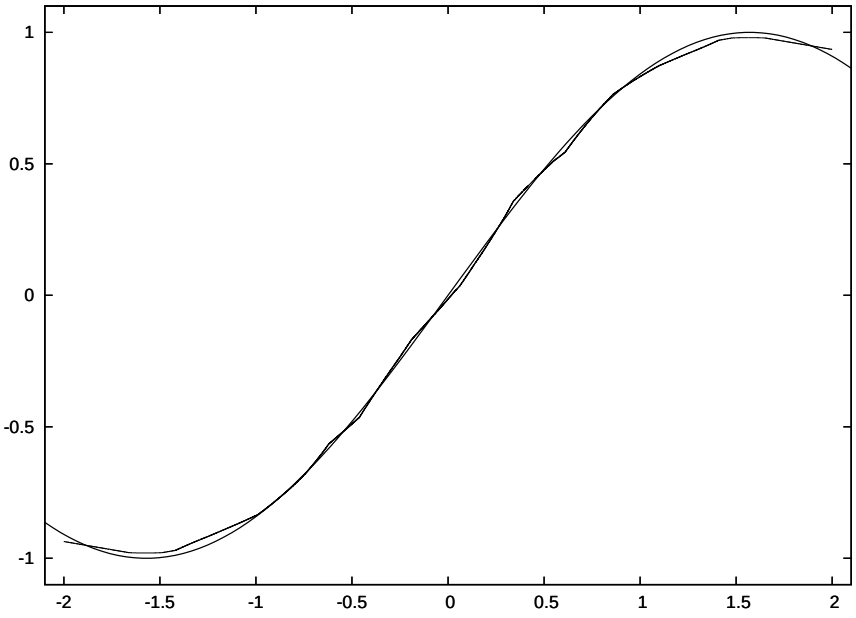

Figure 4: Function obtained for $\beta = 0.8$

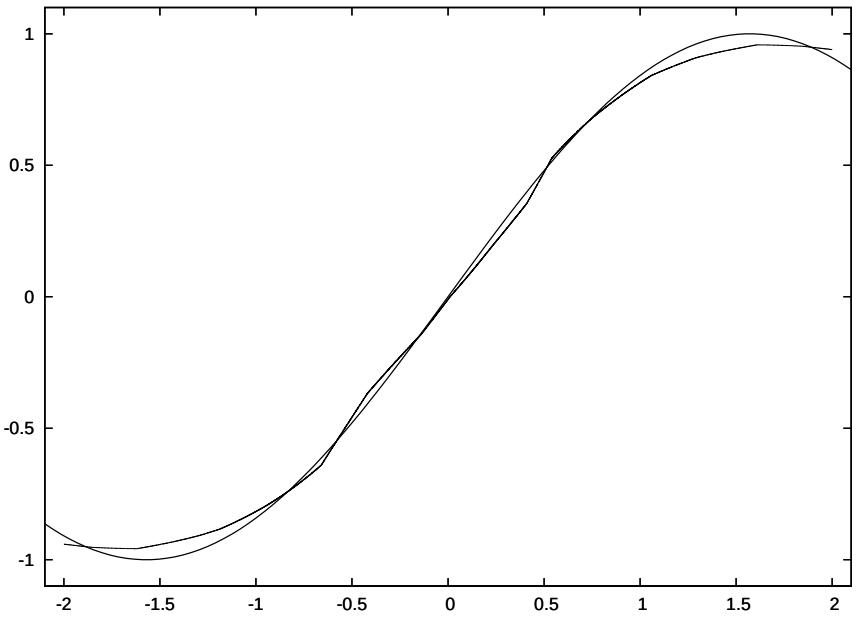

Figure 5: Function obtained for $\beta = 1.2$

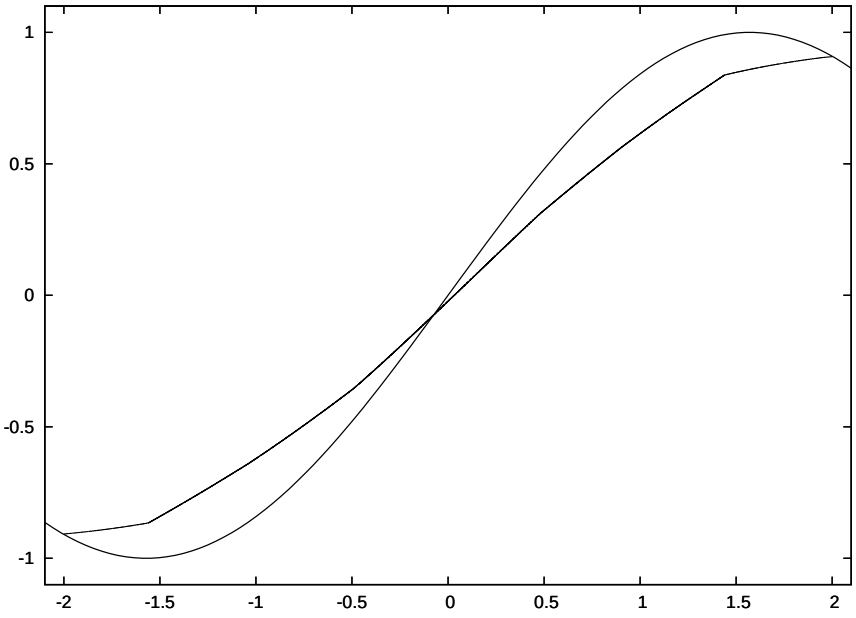

Figure 6: Function obtained for $\beta = 3$

with discontinuity points when switching from one value to another), the fuzzy interpretation we suggested will always have a continuous output (considering that we define the membership degrees on the attributes for each interval to overlap neighbouring intervals).

References

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

[2] Cezary Z. Janikow *Fuzzy Decision Trees: Issues and Methods* 1998

[3] Cristina Olaru, Louis Wehenkel *A Complete Fuzzy Decision Tree Technique* 2003

[4] J.R. Quinlan *C4.5: Programs for Machine Learning* 1993

[5] J.R. Quinlan *Induction on Decision Trees* 1986

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

[7] S. Sestino, T. Dillon. *Using Single-Layered Neural Networks for the Extraction of Conjunctive Rules and Hierarchical Classifications* Journal of Applied Intelligence 1, 1991.