Building Imprecise Classification Trees With Entropy Ranges

Richard J. Crossman
University of Warwick
r.j.crossman@warwick.ac.uk

Joaquin Abellan
University of Granada
jabellan@decsai.ugr.es

Thomas Augustin
University of Munich
augustin@stat.uni-muenchen.de

Frank P.A. Coolen
Durham University
frank.coolen@dur.ac.uk

Abstract

One method for building classification trees is to choose split variables by maximising expected entropy. This can be extended through the application of imprecise probability by replacing instances of expected entropy with the maximum possible expected entropy over credal sets of probability distributions.

Such methods may not take full advantage of the opportunities offered by imprecise probability theory. In this paper, we change focus from maximum possible expected entropy to the full range of expected entropy. We then choose one or more potential split variables using an interval comparison method.

This method is presented with specific reference to the case of ordinal data, and we present algorithms that maximise and minimise entropy within the credal sets of probability distributions which are generated by the NPI method for ordinal data.

Keywords. Imprecise probability, classification trees, nonparametric predictive inference

1 Introduction

The process of classification can be summarised as follows. In a data set D each element is described by n *attribute variables* (or *features*) $X_1, \ldots, X_n$, and a single *class variable* (or *variable in study*) C. The variable X_i takes some value a_i from the set $\mathcal{A}_i$, and the variable C takes some value, or *category*, from $\mathcal{C} = \{c_1, \ldots, c_K\}$. The aim is to take a given vector $\boldsymbol{a} = (a_1, \ldots, a_n)$ and determine the associated category.

One such method is the classification tree. This is a hierarchical graph in which each parent node represents an attribute variable (called the *split variable* of the node), the edges represent the values of that variable, and the leaves represent categories. A data vector $\boldsymbol{a}$ is categorised by starting at the root node and following the appropriate edges until a leaf is reached. The category given at that leaf is the prediction for the associated category of the data point.

Such a method requires finding an order for considering the attribute variables. We base our method upon the one given in [2], summarised as follows:

1. Using information measure IM, calculate $IM(R)$ and $IM(R|X_i)$ (the information measure following splitting on X_i) for each unassigned attribute variable X_i, where R is the data relevant to the current node (i.e. the subset of D which matches the values given to the attribute variables already assigned);
2. If $IM(R|X_{i^*}) := \max_i IM(R|X_i) \leq IM(R)$, go to step 3. Otherwise, split data by the value of X_{i^*}. If the data relevant to the current node is R, then the relevant data for each child node will be $\{\boldsymbol{v} \in R | v_{i^*} = j\}$ where the edge between this node and the child node is labelled $j \in \mathcal{A}_{i^*}$. Return to step 1 for each of these child nodes;
3. This is a leaf node, labelled with the most common class in R. If more than one class is equally common, choose the class most common at the leaf's parent (this approach is due to [4]).

Step 1 will be adapted to make use of imprecise probability, but consider first the information measure when imprecision is not applied. Both $IM(R)$ and each $IM(R|X_i)$ are functions of an associated probability distribution. These distributions are estimated using relative frequencies. For the current data set R, consider each unassigned attribute variable X_i as follows. Define $n^R := |R|$ and denote by n_j^R the number of data points in R with class c_j. Define

$$p_j^R := \frac{n_j^R}{n^R},\ p_j^{\hat{a}_i} := \frac{n_j^{\hat{a}_i}}{n^{\hat{a}_i}} \tag{1.1}$$

where $\hat{a}_i := \{\boldsymbol{v} \in R | X_i = a_i\}$. We also define

$$I(R, X_i) := \sum_{a_i \in \mathcal{A}_i} p(X_i = a_i) H(\boldsymbol{p}^{\hat{a}_i}) \tag{1.2}$$

where $p(X_i = a_i)$ is also estimated using relative frequencies, and where $H(\cdot)$ is Shannon's entropy

$$H(\boldsymbol{p}) = -\sum_{i=1}^{n} p_i \ln(p_i) \tag{1.3}$$

for probability distribution $\boldsymbol{p}$. $H(\boldsymbol{p})$ is maximum when $\boldsymbol{p}$ is uniform, and minimum when $p_i = 1$ for some i.

Define an information measure as follows:

$$IM(R, X_i) := H(\boldsymbol{p}^R) - I(R, X_i). \tag{1.4}$$

Only the $I(R, X_i)$ value determines the next split variable at each node. Further, maximising the information measure is equivalent to minimising $I(R, X_i)$, which in turn requires minimising entropy.

Instead of using the relative frequencies for each X_i to generate a distribution for the categories, we can generate a credal set of probabilites, referred to as a *structure*. This is done in [2] by using the *imprecise Dirichlet model* (IDM) [12], giving the following intervals for $p_j^{\hat{a}_i}$

$$\left[\frac{n_j^{\hat{a}_i}}{n^{\hat{a}_i} + s}, \frac{n_j^{\hat{a}_i} + s}{n^{\hat{a}_i} + s}\right] \tag{1.5}$$

for some value of s, commonly chosen to be 1 or 2.

Alternatives to the IDM exist. In [6] the IDM is replaced with the NPI method for categorical data (requiring a modification to the algorithm, which is contained in that reference). In this paper, the NPI method for ordinal data [7] replaces the IDM. This method takes account of the ordering amongst the categories, resulting (in general) in smaller credal sets than would otherwise be generated. In this set-up categories c_1 and c_K ($c_i,\ i = 2, \ldots, K-1$) are referred to as *boundary* (*internal*) categories. The corresponding component in a probability distribution for a category is referred to as a boundary (internal) component.

A summary of the ordinal NPI method follows. For $X_1, \ldots, X_n, X_{n+1}$ real-valued absolutely continuous and exchangeable random quantities, assume that the first n ordered observed values are denoted by $x_1 < x_2 < \ldots < x_n$, and let $x_0 = -\infty$ and $x_{n+1} = \infty$. We use Hill's assumption $A_{(n)}$ [8] that for a future observation X_{n+1} and for all $j = 1, \ldots, n+1$

$$P(x_{n+1} \in I_j = (x_{j-1}, x_j)) = \frac{1}{n+1}. \tag{1.6}$$

$A_{(n)}$ assumes nothing else, and can be used to define a lower (upper) probability vector $\boldsymbol{L}$ ($\boldsymbol{U}$) for the category of X_{n+1} by a latent variable representation. Assume n observations, with n_j in category c_j. Let Y_{n+1} denote the random quantity representing the category a future observation will belong to. We assume that category c_j is represented by interval Ic_j, where $\cup_{j=1,\ldots,k} Ic_j = \mathbb{R}$ and $Ic_j \cap Ic_i = \emptyset$ for all $i \neq j$. The ordering is such that Ic_j has neighbouring intervals Ic_{j-1} to the left and Ic_{j+1} to the right on the real line, with only one such neighbour when $j \in \{1, k\}$. Assume further that n_j values of $x_1 < x_2 < \ldots < x_n$ are in interval Ic_j. We therefore assume that the event $X_{n+1} \in Ic_j$ is equivalent to the event $Y_{n+1} = c_j$. See [7] for more detail.

The lower probability of a category c_i is therefore equal to the number of intervals I_j entirely contained within Ic_i, and the upper probability of that category is equal to the number of intervals I_j with a non-empty intersection with Ic_i. Hence, the ordinal NPI model replaces (1.5) with:

$$\left[\max(\frac{n_j^{\hat{a}_i} - 1}{n^{\hat{a}_i} + 1}, 0), \frac{n_j^{\hat{a}_i} + 1}{n^{\hat{a}_i} + 1}\right] \tag{1.7}$$

when $1 < j < K$, and otherwise

$$\left[\frac{n_j^{\hat{a}_i}}{n^{\hat{a}_i} + 1}, \frac{n_j^{\hat{a}_i} + 1}{n^{\hat{a}_i} + 1}\right]. \tag{1.8}$$

Therefore all intervals I_j lying entirely within an interval Ic_i are assigned to category c_i, and intervals overlapping both Ic_i and Ic_{i+1} can be assigned entirely to either c_i or c_{i+1}, or split between them.

We can thus talk about the probability mass "on either side" of categories. From this point on the *available mass* to the left (right) of internal category c_j is defined as the probabilty mass that has not been assigned to c_j or c_{j-1} (c_{j+1}) whilst calculating the lower probabilities L_j and L_{j-1} (L_{j+1}). This mass is therefore described as being *available* to $\hat{\boldsymbol{p}}_j$ and $\hat{\boldsymbol{p}}_{j-1}$ ($\hat{\boldsymbol{p}}_{j+1}$). Any distribution $\hat{\boldsymbol{p}}$ within the ordinal NPI structure has the property $\hat{p}_j \geq L_j$; $\hat{p}_j - L_j$ is described as the mass *assigned* to $\hat{p}_j$.

When using either IDM or ordinal NPI, we take the category distribution from each credal set that maximises entropy (note that the distributions of attribute variables are still generated using relative frequencies). This will generate a maximum expected value of each $I(R, X_i)$ and of $H(\boldsymbol{p}^R)$, and so determine our next split variable. However, for $K > 2$ the algorithm given in [2] for maximising entropy does not work within the structure of ordinal NPI (see Section 4), necessitating the algorithm described in this paper.

Once all possible values of (1.4) are non-positive, we do not split further, and decide on the class value to assign by choosing the most common class in R, just as is done in the case without imprecise probability.

However, this application of the IDM or ordinal NPI excludes one of the most fundamental justifications for using imprecise probability: the possibility of circumstances in which we are unable to choose between two options. We therefore describe an alternative method in this paper, which rather than comparing maximum entropies compares the ranges of entropies, and chooses between those ranges only when our method for comparing intervals allows it.

Section 2 defines and explores entropy intervals, which are then used to describe our method, given in Section 3. Section 4 and 5 describe the algorithms by which entropy over the ordinal NPI structure is maximised and minimised, respectively. Section 6 contains conclusions and ideas for future work.

2 Entropy Intervals

Considering entropy intervals requires the following two definitions.

Definition 2.1 For a closed structure $\mathcal{M}$, a vector is defined as a *potential* of $\mathcal{M}$, denoted $\boldsymbol{v}^*$, if $\boldsymbol{v^*} \in \mathcal{M}$ and

$$H(\boldsymbol{v^*}) = \max_{\boldsymbol{w} \in \mathcal{M}} H(\boldsymbol{w}). \tag{2.1}$$

A vector is defined as the *guarantee* of $\mathcal{M}$, denoted $\boldsymbol{v}_*$, if $\boldsymbol{v_*} \in \mathcal{M}$ and

$$H(\boldsymbol{v_*}) = \min_{\boldsymbol{w} \in \mathcal{M}} H(\boldsymbol{w}). \tag{2.2}$$

If $\frac{1}{k}(1, \ldots, 1) \in \mathcal{M}$, then $v^* = \frac{1}{k}(1, \ldots, 1)$. Any probability vector in $\mathcal{M}$ with a component equal to 1 is a guarantee of $\mathcal{M}$.

The names of these properties are justified as follows: for a given X_i the entropy of the potential and guarantee generate respectively the maximum and minimum value of $I(R, X_i)$. Thus we can guarantee a minimum value for this function, but also talk of the potential maximum. This is also true for $H(\boldsymbol{p}^R)$.

In a convex structure, the potential is unique (see the algorithm in [2]). This is not necessarily true of the guarantee; when $\mathcal{M} = [0, 1] \times [0, 1]$, both $(1, 0)$ and $(0, 1)$ are guarantees.

Because entropy is a continuous function, and the ordinal NPI structure is connected, we can define an entropy interval as follows.

Definition 2.2 The *entropy interval* of a connected structure $\mathcal{M}$ is defined as

$$\{H(\boldsymbol{v}) : \boldsymbol{v} \in \mathcal{M}\} = [H(\boldsymbol{v}_*), H(\boldsymbol{v}^*)]$$

where $\boldsymbol{v}_*$ and $\boldsymbol{v}^*$ are the guarantee and the potential of $\mathcal{M}$ respectively.

Example 4.1 Consider $K = 8$ classes, and six observations (1,0,0,2,0,3,0,0). From [7] we have that the structure is contained within the following set:

$$\frac{1}{7}([1, 2], [0, 1], [0, 1], [1, 3], [0, 1], [2, 4], [0, 1], [0, 1]). \tag{2.3}$$

The maximum entropy algorithm adapted for ordinal data (see Section 4) gives the following vectors at each stage

1. $\frac{1}{7}(1, 0, 0, 1, 0, 2, 0, 0)$, 2. $\frac{1}{7}(1, 0, 0, 1, 1, 2, 0, 0)$
3. $\frac{1}{14}(2, 1, 1, 2, 2, 4, 0, 0)$, 4. $\frac{1}{14}(2, 1, 1, 2, 2, 4, 1, 1)$

and the minimum entropy algorithm (see Section 5) gives

1. $\frac{1}{7}(1, 0, 0, 1, 0, 2, 0, 0)$, 2. $\frac{1}{7}(1, 0, 0, 1, 0, 4, 0, 0)$
3. $\frac{1}{7}(2, 0, 0, 1, 0, 4, 0, 0)$

The resulting entropy interval is $[0.9557, 1.9459]$. For comparison, note that the full entropy range for an 8 element probability distribution is $[0, 2.079]$, and that had we used the algorithm given in [2] without taking into account the structure of the model, we would have incorrectly generated a potential with entropy 1.9668. This concludes the example.

Instead of a single value $I(R, X_i)$, consider an interval $[\underline{I}(R, X_i), \overline{I}(R, X_i)] := I_i$ where the bounds of the interval are calculated using guarantees and potentials in the obvious way. Further, replace $H(\boldsymbol{p}^R)$ with the interval I_R, generated by the guarantee and potential of the current data set R.

These intervals provide an alternative method for choosing the split variables. Define a set of intervals $\mathcal{I} = \{I_{a_1}, \ldots, I_{a_n}\}$, where each a_i corresponds to a potential split variable X_{a_i}. Remove from $\mathcal{I}$ any interval that is dominated by another interval in the set. There are various methods by which one can determine dominance, but in this paper we use the simplest: *interval dominance*. Under this method, interval $I_i = [c_i, d_i]$ dominates interval $I_j = [c_j, d_j]$, denoted $I_i >_{\mathrm{d}} I_j$, iff $c_i \geq d_j$. The use of alternative methods for comparing intervals [11] can be explored, this is left as a topic for future research.

Once all dominated intervals have been removed, we say we cannot choose between each of variables corresponding to the remaining elements of $\mathcal{I}$ as the next choice for the split variable.

3 Imprecise classification trees

Just as imprecise probabilities are expressed as sets rather than single values, an imprecise classification tree is expressed as a forest.

Consider node P at the end of a path, length l, from the root node. There are $n - l$ choices for the next split variable, denoted $X_{P_1}, X_{P_2}, \ldots, X_{P_{n-l}}$. If no interval I_{P_j}

dominates I_R, then there is no split (as no split variable can be considered superior to no split at all). Otherwise, create a set $\mathcal{S}$ as follows

$$\mathcal{S} := \{X_{P_j} | \exists \text{ no } i \neq j \text{ s.t. } I_{P_i} >_{\mathrm{d}} I_{P_j}\}. \qquad (3.1)$$

Therefore $\mathcal{S}$ is the set of all potential split variables for which a superior choice of split variable cannot be found. Let $m := |\mathcal{S}|$. We create $m-1$ identical copies of the current tree. This produces m trees, each of which uses a different split variable, chosen from $\mathcal{S}$ to continue the current path.

If it is determined that no split variable is preferable to not splitting, the node becomes a leaf. The method in [2] uses the relative frequency of the current data set to choose the most likely category (if two or more categories are equally likely, the most likely category at the parent node is chosen). One alternative would be to use the NPI method for ordinal data to construct a structure for the category probabilities.

A method by which the structures of the trees can be combined is now required. All trees should not be given equal weight, or some choices for split variables may dominate others. For example, consider three Boolean attribute variables X_1, X_2, X_3. Variables X_1 and X_2 are chosen for the initial split, so X_i is chosen as the root node for Tree i, with $i = 1, 2$. In Tree 1, whatever the value of X_1, we split on X_2 next. In Tree 2, when $X_2 = 0$, we split on X_1 next, but when $X_2 = 1$, we cannot choose between splitting next on X_1 or X_3. Therefore Tree 2 splits upon X_1, and a new tree is generated, Tree 3, which splits upon X_3. No further splits are made (see Figure 1).

Giving each of these three trees equal weight would imply that the initial choice of X_2 is twice as desirable as the choice of X_1. There is no justification for this.

There are various ways to tackle this issue. We could weight each tree according to the nature of its relevant entropy interval: how wide it is, and how far it lies from the interval for the full data set which it is dominating. For now, however, each tree is given weight one, which decreases each time there is a split after the root node. In the situation shown in Figure 1, Tree 1 would be given a weight of 1, and Trees 2 and 3 a weight of $\frac{1}{2}$ each, ensuring each choice of root node is given equal weight. This method is equivalent to creating duplicate trees. For example, in the situation shown in Figure 1, rather than weight each tree, Tree 1 could be duplicated.

There are many potential methods by which such a forest can be used to classify a data point. Denote by $\mathcal{X}$ the set of all categorisations given by the forest. The most conservative approach would be to simply return $\mathcal{X}$, making the imprecise tree a credal classifier (see e.g. [13]). Alternatively, for each c_i we can sum the weights of the trees which returned c_i, denoted Σ_i and choose c_* where $\Sigma_* = \max_i \Sigma_i$, selecting randomly amongst all categories for which $\Sigma_i = \Sigma_*$ if the maximum is non-unique. A third option is to return each category c_i for which $\Sigma_i \geq C$, for some constant C. Setting $C = 0$ ($C = \Sigma_*$) reduces to the first (second) method. These approaches will be compared in a later paper.

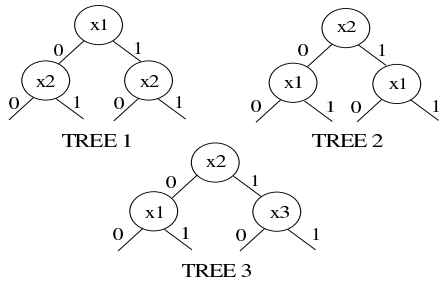

Figure 1: Forest generated by this method

Example 5.1 A small table of data, shown in Table 2, is used to draw an imprecise classification tree. Note that this example is included to illustrate our method, not to consider its efficacy. Indeed, the binary categories in this data set are categorical, not ordinal. Since $K = 2$, we can use the maximum entropy algorithm given in [2] (see Section 5 for the minimum entropy algorithm). We use the first forty data points as a training set, and the final ten as a test set. The binary nature of the category set makes it easy to calculate entropies:

$$H(\boldsymbol{v}) = -\frac{n_1^R}{a}\ln(\frac{n_1^R}{a}) - \frac{n_2^R}{a}\ln(\frac{n_2^R}{a}) \qquad (3.2)$$

where n_i^R is the number of instances of category i and $a := n_1^R + n_2^R$.

The method generates three trees, displayed in Figures 3 to 5. Tree 1 has weight 1, the others have weight $\frac{1}{2}$. Each leaf is labelled with the category assigned to it. Note that it appears in some cases that the set of possible values of the attribute variables changes from tree to tree. This is because, depending on previous attribute variables, the set of data R under consideration may not contain any instances of one or more values of the variable chosen as the split variable.

It is simple to compare this method with the one given in [2], as that method generates only one tree, which is in fact Tree 2. Tree 2 is correct 8 times out of 10. The imprecise tree is correct every time, though it returns both categories on two occasions; this is true irrespective of the choice of C. There is a one-to-one correspondence between the data points incorrectly classified by Tree 2, and the data points for which the imprecise tree gives both categories.

4 Maximum entropy algorithm

In this section the algorithm in [2] is adapted for the ordinal NPI method. We will provide an example later in the section demonstrating why that algorithm cannot be applied to the ordinal NPI case directly, but in short, it fails because it requires that the structure be convex, which is not the case here.

Our algorithm is too complex to be described in full, instead an overview is presented, including the relevant lemmas and proofs. This complexity is required despite the

fact that the structure and the function to be maximised are both simple in form, because the constraints upon the maximisation problem are complicated by the conditions imposed by the ordinal NPI approach. For example, whilst we have that $L_i \leq p_i \leq U_i$ and $L_{i+1} \leq p_{i+1} \leq U_{i+1}$, we also have that $L_i + L_{i+1} + \frac{1}{n+1} \leq p_i + p_{i+1} \leq U_i + U_{i+1} - \frac{1}{n+1}$. Moreoever, the sum of three adjacent elements will have its own constraints, and so on.

Maximum Entropy Algorithm

This algorithm is broadly similar to one presented in [6], which utilises NPI for the categorical case. Our consideration of the probability mass being available "on either side" of a component is based on the approach in that reference.

The algorithm requires two K-vectors $\boldsymbol{v}^L := (0, 1, \ldots, 1)$ and $\boldsymbol{v}^R := (1, \ldots, 1, 0)$. The j-th component of $\boldsymbol{v}^L$ ($\boldsymbol{v}^R$) represents the amount of mass available to $\hat{p}_j$ to the left (right). These vectors are updated after each mass assignment.

The algorithm can be broken down into two processes. The first process assigns the mass between two components in situations in which only those two components need be considered. For example, for $j \in \{1, \ldots, K-2\}$, let

$$|\hat{p}_j - \hat{p}_{j-1}| \geq \frac{2}{n+1}, \tag{4.1}$$

and assume without loss of generality (WLOG) that $\hat{p}_j > \hat{p}_{j-1}$. The convex nature of the entropy function means entropy is maximised by adding mass to the smallest components of $\hat{p}$ possible. Even if all mass to the left of c_{j-1} is assigned to $\hat{p}_{j-1}$, then that component will be no larger than $\hat{p}_j - \frac{1}{n+1}$. This means we must assign all mass between c_j and c_{j-1} to $\hat{p}_{j-1}$. An exception is the case where $n_{j-1} = n_{j-2} = 0$. In this case we cannot be sure that assigning all mass to $\hat{p}_{j-1}$ is justified. We do however know that no mass to the right of c_j can be assigned to $\hat{p}_j$. An extension of this argument can be applied to the boundary components and their neighbours.

Further, if there exists any adjacent components for which $|\hat{p}_j - \hat{p}_{j-1}| = \frac{1}{n+1}$, v^R_{j-1} and v^L_j are non-zero, there may be another assignment to be made. If we assume WLOG that $\hat{p}_{j-1} < \hat{p}_j$, then if $v^L_{j-1} = 0$, we must assign the mass between c_j and c_{j-1} to $\hat{p}_{j-1}$. Again, in situations with consecutive categories with zero observations, we may at this stage be only able to decide that some components *cannot* be assigned available mass, without being able to decide which components *should* be assigned that mass. Lastly, if $\hat{p}_j = \hat{p}_{j-1} = 0$ and $v^R_j = v^L_{j-1} = 0$, the mass is shared equally between the components.

The second process can consider three or more components simultaneously, using the concept of *strings*.

Definition 4.1 The categories $c_a, c_{a+1}, \ldots, c_b$ form a string if $v^R_j + v^L_j > 0$ for all $a \leq j \leq b$ and further $v^R_{a-1} = 0$ when $c_a \neq c_1$ and $v^L_{b+1} = 0$ when $c_b \neq c_K$.

We define $\mathcal{S} := \{c_a, c_{a+1}, \ldots, c_b\}$, and refer to the vector $(\hat{p}_a, \hat{p}_{a+1}, \ldots, \hat{p}_b)$ as the *string vector*. The *length* of a string equals the number of classes in the string.

The algorithm finds a string within the vector $\hat{\boldsymbol{p}}$ (which might be the entire vector), assigns mass to either reduce the length of the string or split it in two, and then finds another string, until none remain. By definition there cannot be more than $[\frac{K}{2}]$ strings, each of maximum length K, so the number of iterations required to assign all available mass must be less than $\frac{K(K+1)}{2}$.

The algorithm makes use of the following result.

Lemma 4.1 Let $\{c_1, \ldots, c_K\}$ contain the set of strings $\zeta := \{S_i,\ i = 1, \ldots, r\}$. String S_i has length l_i and contains categories c_{a_i} to c_{b_i}; $a_{i+1} > b_i$. Categories with their mass assignments already determined will belong to no S_i. The vector maximising entropy in the structure is uniquely determined by the vector maximising entropy over ζ. Let $\boldsymbol{w}^i$ represent the observations $(n_{a_i}, \ldots, n_{b_i})$. Consider $\boldsymbol{w}^i$ as a complete observation vector, and generate $\boldsymbol{v}^i$ as the corresponding vector maximising entropy. Entropy over ζ is maximised by the vector $d(t_1\boldsymbol{v}^1, t_2\boldsymbol{v}^2, \ldots, t_r\boldsymbol{v}^r)$ for $t_i = \frac{l_i+1}{K+1}$ and normalising constant d.

Proof.

$$H(\hat{\boldsymbol{p}}) = -\sum_{j=1}^{K} \hat{p}_j \ln(\hat{p}_j) = -\sum_{j:c_j \in \zeta} \hat{p}_j \ln(\hat{p}_j) + C$$

where $C = -\sum_{j:c_j \notin \zeta} \hat{p}_j \ln(\hat{p}_j)$ is constant. Therefore maximising $H(\hat{\boldsymbol{p}})$ is equivalent to maximising $-\sum_{j:c_j \in \zeta} \hat{p}_j \ln(\hat{p}_j)$. Moreover, when considering the maximum algorithm for $\tilde{S}_i$,

$$\begin{aligned} \frac{H(\hat{\boldsymbol{p}})}{m} + \frac{\ln(m)}{m} &= -\sum_{k=1}^{l_i} \left(\frac{\hat{p}_k}{m}(\ln(\hat{p}_k) - \ln(m))\right) \\ &= -\sum_{k=1}^{l_i} \left(\frac{\hat{p}_k}{m}(\ln(\frac{\hat{p}_k}{m}))\right) \\ &= H(\frac{\hat{\boldsymbol{p}}}{m}) \end{aligned} \tag{4.2}$$

where $m > 0$ is constant. The final expression in (4.2) is a slight abuse of notation, since entropy is generally only considered in terms of probability distributions, but there is no mathematical problem with considering it as a function over all l_i-vectors with non-negative elements. We define by V^i_a the set of all l_i-vectors with non-negative elements which sum to a (hence V^K_1 is the set of all possible probability distributions over the categories), and define by H_{S_i} the contribution to the overall entropy supplied by the string S_i. This means that 4.2 leads to

$$\begin{aligned} H(\boldsymbol{v}^*) &= \max_{\boldsymbol{v} \in V^i_1} H(\boldsymbol{v}) = t_i(\max_{\boldsymbol{v} \in V^i_{t_i^{-1}}} H(\boldsymbol{v})) - \ln(t_i) \\ &= t_i(H(\frac{\boldsymbol{v}^*}{t_i})) - \ln(t_i) \end{aligned} \tag{4.3}$$

so the vector which maximises entropy over $\tilde{S}_i$ is a positive multiple of the vector which maximises the contribution

of string S_i to the whole vector. Hence

$$\begin{aligned}\max H(\boldsymbol{p}) &= \max(-\sum_{j:c_j\in\zeta} p_j \ln(p_j)) + C \\ &= \sum_i^r (t_i(\max_{\boldsymbol{v}\in V_{t_i^{-1}}^i} H_{S_i}(\boldsymbol{v})) - \ln(t_i)) + C_2 \\ &= \sum_i^r t_i(\max_{\boldsymbol{v}\in V_{t_i^{-1}}^i} H_{S_i}(\boldsymbol{v})) + C \\ &= \sum_i^r t_i(H_{S_i}(\boldsymbol{v}^i)) + C_2 \qquad (4.4)\end{aligned}$$

where the inclusion of the t_i is justified by the need to rescale each vector $\boldsymbol{v}^i$, and C_2 is constant. This completes the proof. □

From Lemma 4.1 each string can be considered independently, as though its corresponding observation vector was the only one under consideration, with c_{a_i} and c_{b_i} as boundary categories. Let $x_1 := \min_{j\in\mathcal{S}} \hat{p}_j$ and $x_2 := \max_{j\in\mathcal{S}} \hat{p}_j$. In each case, a mass assignment is made and the vectors $\boldsymbol{v}^L$ and $\boldsymbol{v}^R$ are updated accordingly.

There are nine different types of string, all of which the algorithm deals with differently. A full description of these methods will be presented in a later paper; the methods described below are by no means exhaustive and are merely intended as a demonstration of the ideas involved.

We aim to ensure the components of a string are as close to being equal as is possible. If $x_2 > x_1 + \frac{1}{n+1}$, equality is impossible. However, in this case we can denote by y_1 and $y_2 > y_1$ the smallest integers for which $\hat{p}_{y_1} \in \{x_1, x_2\}$, $y_2 = I_{[\hat{p}_{y_1}=x_2]}x_1 + I_{[\hat{p}_{y_1}=x_1]}x_2$, and for which $\hat{p}_j \notin \{x_1, x_2\}$ for all $y_1 < j < y_2$ (where I_A is the indicator function). We have that $\min(\hat{p}_{y_1}, \hat{p}_{y_2}) = x_1$ and $\max(\hat{p}_{y_1}, \hat{p}_{y_2}) = x_2$. Moreover, for any $\min(y_1, y_2) < j < \max(y_1, y_2)$, $x_1 < \hat{p}_j < x_2$.

Assume WLOG that $p_{y_1} = x_1$, and that $x_1 > 0$ (if $x_1 = 0$ we require a different approach, not described here). The mass between c_{y_1} and c_{y_1-1} must be available, and even if this mass is assigned entirely to $\hat{p}_{y_1}$, $\min\{\hat{p}_{y_1}, \ldots, \hat{p}_{y_2}\} = \hat{p}_{y_1}$ holds. Each component between $\hat{p}_{y_1}$ and $\hat{p}_{y_2}$ therefore requires at least $\frac{1}{n+1}$ mass to reach the value of $\hat{p}_{y_2}$, as does $\hat{p}_{y_1}$ itself, so all available mass will be assigned before $\hat{p}_{y_2}$ is eligible to receive any of it. The mass between c_{y_2-1} and c_{y_2} is therefore assigned to $\hat{p}_{y_2-1}$.

In the case where $x_2 = x_1 + \frac{1}{n+1}$, we denote by y_1 the number of components equal to x_1, and by y_2 the number equal to x_2. Therefore, in this situation, we would want to use a total mass $\frac{y_1}{n+1}$ to increase all minima from x_1 to x_2. We would then share the remaining $\frac{y_2}{n+1}$ mass equally between all components.

This mass assignment is not always allowed in our NPI structure, as shown by the example below.

Consider the situation defined by the observation vector $(2, 2, 3, 2, 2, 2, 3, 2, 2, 2, 2)$, with associated lower probability vector $\frac{1}{25}(2, 1, 2, 1, 1, 1, 2, 1, 1, 1, 2)$. Entropy would be maximised by increasing each element equal to $\frac{1}{25}$ by $(\frac{1}{25})(\frac{14}{11}) = \frac{14}{275}$, and each increasing element equal to $\frac{2}{25}$ by $(\frac{1}{25})(\frac{3}{11}) = \frac{3}{275}$. Note that this is also the assignment given by the algorithm in [2]. Figure 1 shows this assignment is not possible.

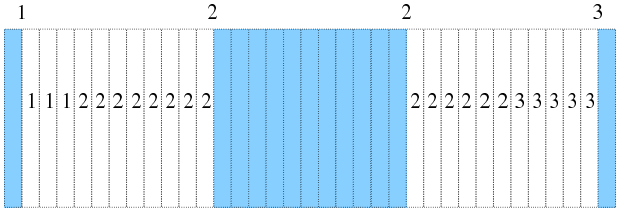

Figure 2: Inability to apply Abellan/Moral algorithm

Figure 1 shows the available mass between the pairs of categories c_1 and c_2, and c_2 and c_3. Each of these pieces of available mass are of size $\frac{1}{25}$, and each of the smaller rectangles has mass $\frac{1}{275}$. It is clear from the diagram that assigning enough probability mass to $\hat{p}_1$ and $\hat{p}_2$ to make them both equal to $\frac{25}{275}$ forces the value of $\hat{p}_3$ to become at least $\frac{27}{275}$.

Therefore the available probability mass cannot be spread across $\hat{\boldsymbol{p}}$ so as to produce a uniform distribution. The available mass that must be assigned to categories c_1, c_2 and c_3 is too great. We therefore require not only a method for assigning mass that is compatible with our model, but a proof demonstrating that the resulting distribution does indeed give the maximum entropy possible.

This approach is summarised as follows. If $x_2 = x_1 + \frac{1}{n+1}$, we try to assign the uniform distribution $\frac{1}{n}(1, \ldots, 1)$, category by category, from left to right, where n is a function of the number of categories within the string with associated components equal to x_1 (when $x_1 = x_2$, this assignment is automatically successful). This assignment will fail if either all available mass to some category c_i is assigned without $\hat{p}_i$ reaching $\frac{1}{n}$, or $\hat{p}_i$ reaches $\frac{1}{n}$ and yet mass remains available which must be assigned to category c_i.

Once either event occurs (if neither does, then the uniform distribution can in fact be reached), we start again from $\hat{p}_{c_i}$ and attempt the same mass assignment. This may occur several times. As a result, we will have divided the components in the string vector into two or more sets. Each set has either too much or too little available mass for the desired assignment. Moreover, there must exist sets $\{c_{a_1}, \ldots, c_{b_1}\}$ and $\{c_{b_1+1}, \ldots, c_{b_2}\}$ for which one has too much mass, and the other too little. In such a situation, the algorithm can create two new strings, one ending with c_{b_1} and another beginning with c_{b_1+1}. The justification for this follows from Lemma 4.2.

Lemma 4.2 Let $0 < l_1 < n$ and $l_2 = n - l_1$. Let $\mathcal{M}$ be a closed credal set of n-vectors for which $\boldsymbol{u} \in \mathcal{M} \Leftrightarrow \boldsymbol{u} = (\boldsymbol{v}, \boldsymbol{w})$, where $\boldsymbol{v}$ is a l_1-vector and $\boldsymbol{w}$ is a l_2-vector with the following properties: $\sum_{i=1}^{l_1} v_i > k_1$ and $\sum_{j=1}^{l_2} w_j < k_2$, and $\frac{k_1}{l_1} > \frac{k_2}{l_2}$. Then if $(\frac{k_1}{l_1}(1, \ldots, 1), \frac{k_2}{l_2}(1, \ldots, 1))$ lies within $\mathcal{M}$, it is the distribution which maximises entropy.

Proof. From Lemma 4.1 entropy orderings are invariant

-	O-NPI	IDM	IGR	IG	W-L
O-NPI	- -	(6,14) (0,4)	(11,9) (1,4)	(11,9) (2,4)	-4 -9
IDM	(14,6) (4,0)	- -	(14,6) (3,2)	(11,9) (1,1)	18 5
IGR	(9,11) (4,1)	(6,14) (2,3)	- -	(10,10) (0,0)	-10 2
IG	(9,11) (4,2)	(9,11) (1,1)	(10,10) (0,0)	- -	-4 2

Table 1: Comparison of methods

over multiplication by a constant, so we can assume $k_1 + k_2 = 1$. Also by Lemma 4.1, following the assignment of mass α to $\boldsymbol{v}$ and mass $1-\alpha$ to $\boldsymbol{w}$, the entropy is maximised by setting $v_i = \frac{\alpha}{l_1}$, $\forall i$ and $w_j = \frac{1-\alpha}{l_2}$, $\forall j$. Clearly, $\alpha \geq k_1$ and $1-\alpha \leq k_2$. The proof, then, reduces to demonstrating that, indeed, entropy is maximised by setting $\alpha = k_1$.

Construct a structure for which $\boldsymbol{L} = (\boldsymbol{v}_L, \boldsymbol{w}_L)$ and $\boldsymbol{U} = (\boldsymbol{v}_U, \boldsymbol{w}_U)$, where $(\boldsymbol{v}_L)_i = \frac{k_1}{l_1}$ and $(\boldsymbol{v}_U)_i = 1$ $\forall i$, and $(\boldsymbol{w}_L)_j = 0$ and $(\boldsymbol{w}_U)_j = \frac{k_2}{l_2}$ $\forall j$. By the algorithm in [2], the elements corresponding to $\boldsymbol{w}$ must all reach their upper bound before the elements of $\boldsymbol{v}$ are considered at all. Therefore, giving more mass to $\boldsymbol{v}$ than the minimum requirement violates the algorithm, and so the given assignment must, indeed, maximise entropy. □

Lemma 4.2 proves that when $\{c_{a_1}, \ldots, c_{b_1}\}$ has too much (too little) mass and $\{c_{a_2}, \ldots, c_{b_2}\}$ has too little (too much) mass for the uniform distribution to be assigned, entropy cannot be increased by taking mass from the set of categories with less mass, so long as both sets can separately be assigned mass in such a way as to make all components equal. However, if such an assignment is not possible, we can simply split the set up again, and once again apply Lemma 4.2, and so on. This justifies splitting the string as described.

We now compare our method with the IDM imprecise method, along with the Info Gain [9] and Info Gain Ratio methods [9] over 21 data sets in which the categories can plausibly be argued to be ordinal. All these methods were run using the computer package known as WEKA. The results are given in Table 1. The first pair of numbers in each cell represent the number of wins and losses for the row classifier with respect to the column classifier, in terms of percentage of correct classification. For example, the pair (12,6) in the second row tells us that the IDM classifier outperformed the O-NPI classifier 12 times, and was outperformed 6 times. The second pair of numbers also represent wins and losses, this time using a paired t-test at the 5% level. The final row of the table gives the total number of wins minus the total number of losses for both tests.

These results do not show any obvious improvement in replacing the IDM structure with that of ordinal NPI. Indeed, it seems to be performing worse than the IDM method, and roughly equivalently to the IGR and IG methods.[1]

We now describe the algorithm for minimising entropy.

5 Minimum entropy algorithm

Minimising entropy is difficult in general. However, in the specific case of ordinal NPI, it is quite simple. We begin with three lemmas.

Lemma 5.1 When minimising entropy, all available mass between observed categories is assigned entirely to one of the associated components.

Proof. Let $H_2(v_1, v_2) = -v_1 \ln(v_1) - v_2 \ln(v_2)$ be the contribution of two components to the entropy. The entropy function is concave, so if $b \leq c$ we have

$$H_2(a+c, a) \leq H_2(a+c-b, a+b). \tag{5.1}$$

Therefore for any values a and c, $H_2(\cdot,\cdot)$ is minimised when either $b = 0$ or $b = c$. Let the adjacent observed categories have components $\hat{p}_i = a$ and $\hat{p}_j = a+c$, and let the mass between them be denoted by $m > 0$. Then from (5.1) we have two inequalities

$$\begin{aligned} H_2(a+c+m, a) &\leq H_2(a+c+m-b, a+b) \\ H_2(a+c, a+m) &\leq H_2(a+c-b, a+b+m), \end{aligned}$$

meaning the minimum entropy occurs when the entirety of m is assigned either $\hat{p}_i$ or $\hat{p}_j$. □

Lemma 5.2 When minimising entropy, no unobserved category is assigned mass.

Proof. Setting $a = 0$ in (5.1) shows any assignment of mass to a zero component leads to an increase in entropy compared to assigning that mass to a non-zero component. Therefore this should never be done if an alternative is available, which is always the case for unobserved categories in the ordinal NPI case. □

Therefore the minimum entropy algorithm can operate simply by assigning all unassigned mass between observed categories c_i and c_j entirely to $\hat{p}_i$ or to $\hat{p}_j$.

Lemma 5.3 When minimising entropy, for any two components $\hat{p}_i$ or $\hat{p}_j$ corresponding to adjacent observed internal categories, the mass between c_i and c_j is assigned to $\hat{p}_i$ if $\hat{p}_i > \hat{p}_j$.

Proof. Consider any pair of adjacent observed internal categories c_j, c_{j+1} for which $\hat{p}_j \neq \hat{p}_{j+1}$. Assume WLOG that $\hat{p}_j < \hat{p}_{j+1}$, and set $\hat{p}_j =: r_2$ and $\hat{p}_{j+1} =: r_3 = r_2 + \frac{1}{n+1} + \alpha$ for some $\alpha \in \mathbb{N}$. Since both categories are internal, we can also consider the components $\hat{p}_{j-1} =: r_1$ and $\hat{p}_{j+2} =: r_4$. Between these four categories is available mass $\frac{3}{n+1}$, and from Lemma 5.1 we have that three (not necessarily distinct) components must receive $\frac{1}{n+1}$ mass.

[1] Note that the WEKA code for the exact O-NPI algorithm is still in development.

There are eight ways that this mass assignment can be carried out. These can be divided into four pairs. In each pair one assignment gives the mass between $\hat{p}_j$ and $\hat{p}_{j+1}$ to $\hat{p}_j$, and in the other it gives it to $\hat{p}_{j+1}$; the other two mass assignments are identical. The eight mass assignments are given in the table below, along with the resulting size of each component. Each pair is presented together, and the first case in the pair is always the one in which the smaller component is assigned the mass.

-	r_1	r_2	r_3	r_4
1	r_1	$r_2 + \frac{2}{n+1}$	$r_2 + \frac{1}{n+1} + \alpha$	$r_4 + \frac{1}{n+1}$
2	r_1	$r_2 + \frac{1}{n+1}$	$r_2 + \frac{2}{n+1} + \alpha$	$r_4 + \frac{1}{n+1}$
3	r_1	$r_2 + \frac{2}{n+1}$	$r_2 + \frac{2}{n+1} + \alpha$	r_4
4	r_1	$r_2 + \frac{1}{n+1}$	$r_2 + \frac{3}{n+1} + \alpha$	r_4
5	$r_1 + \frac{1}{n+1}$	$r_2 + \frac{1}{n+1}$	$r_2 + \frac{1}{n+1} + \alpha$	$r_4 + \frac{1}{n+1}$
6	$r_1 + \frac{1}{n+1}$	r_2	$r_2 + \frac{2}{n+1} + \alpha$	$r_4 + \frac{1}{n+1}$
7	$r_1 + \frac{1}{n+1}$	$r_2 + \frac{1}{n+1}$	$r_2 + \frac{2}{n+1} + \alpha$	r_4
8	$r_1 + \frac{1}{n+1}$	r_2	$r_2 + \frac{3}{n+1} + \alpha$	r_4

We now prove that for all four pairs, the second assignment has lower entropy than the first. Therefore, irrespective of how the mass between $\hat{p}_{j-1}$ and $\hat{p}_j$ and between $\hat{p}_{j+1}$ and $\hat{p}_{j+2}$ is assigned, we must assign the mass between $\hat{p}_j$ and $\hat{p}_{j+1}$ to the larger component.

This is immediately clear for the second, third and fourth pairs by the concave nature of the entropy function; entropy is minimised by setting two values as far apart as possible. For the first pair, the two cases are equivalent when $\alpha = 0$, and we still lose nothing by assigning the mass to the larger component. If $\alpha \geq 1$, we can define $\alpha - \frac{1}{n+1} =: \alpha_1 \geq 0$ and re-write the pair

-	r_1	r_2	r_3	r_4
1	r_1	$2_1 + \frac{2}{n+1}$	$r_2 + \frac{2}{n+1} + \alpha_1$	$r_4 + \frac{1}{n+1}$
2	r_1	$2_1 + \frac{1}{n+1}$	$r_2 + \frac{3}{n+1} + \alpha_1$	$r_4 + \frac{1}{n+1}$

and once again from the fact that the entropy function is concave we see that minimising entropy requires assigning the mass to the larger component.

We have then that for all mass assignments between the category pairs c_{j-2}, c_{j-1} and c_j, c_{j+1}, entropy is minimised by adding the available mass to the larger component. Therefore every individual slice of available mass between c_j and c_{+1} can be considered separately, so long as both neighbouring categories are internal, and $\hat{p}_j \neq \hat{p}_{j+1}$. □

Boundary categories are handled in a similar way. If a boundary category is unobserved, no mass will be assigned to it. Otherwise, we can consider, say, $\hat{p}_1$ as being equivalent to an internal category for which the mass on the left has already been assigned elsewhere. This allows us to make use of Lemma 5.3.

Lastly we deal with situations in which there are consecutive equal components. Suppose there are n consecutive components all of size m. If $n = 2$, we can assign the mass to either component. If $n = 3$, we assign all mass to the central component, as $H(m + a, m, m + a) > H(m, m + 2a, m)$. This leaves the third component unchanged, and so we can use this more generally to reduce the number of consecutive components from n to $n - 2$. This means we can continually re-apply this assignment until either all mass has been assigned, or we are left with just two equal components with available mass between them. We then simply assign that mass to either side.

Note that our algorithm runs from left to right, assigning mass to the larger component each time, and once finished, returns and deals with each sequence of equal components. Running the algorithm from right to left might result in a different vector being returned. In other words, the vector we find results in a global minimum for entropy, but it does not follow that no other vector could not also produce a global miniminum for entropy. As an obvious example, consider $K = 3$ and observation vector (1,0,1). Clearly $\boldsymbol{L} = \frac{1}{3}(1, 0, 1)$ and $\boldsymbol{U} = \frac{1}{3}(2, 1, 2)$. Our minimisation algorithm returns the vector $\frac{1}{3}(2, 0, 1)$, but clearly the vector $\frac{1}{3}(1, 0, 2)$ will have an identical entropy value.

6 Conclusions and Further Work

At each stage of tree construction the method presented here allows for the possibility that we cannot choose between potential split variables. This has been considered previously regarding choice of root node [4], but we are aware of no method in which this idea is applied to the construction of the whole tree, or one which compares entropy ranges. Clearly, it remains to test this method against others - at the time of writing the WEKA code for our method has not yet been written - but by expanding focus beyond the root nodes and by utilising comparisons between intervals, this method combines classification and imprecise probability in an attractive way, by recognising situations in which it is unreasonable to consider one split variable choice as clearly superior to another.

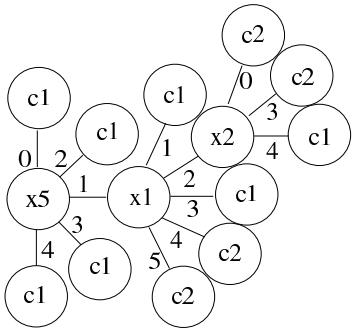

Figure 3: Example 5.1: Tree 1

Further work is required on considering how best to collate the set of categories given by the imprecise decision tree for each data point. It would also be of value to consider more thoroughly the implications of minimising entropy, particularly with regard to unobserved categories. One of the criticisms regarding attempts to maximise entropy is that it invariably gives as much mass as possible to categories that were unobserved. This is difficult to justify

	A_1	A_2	A_3	A_4	A_5	C
1	1	3	0	0	2	1
2	1	3	0	1	1	1
3	1	2	0	1	0	1
4	1	2	0	1	0	1
5	1	3	0	1	0	1
6	1	3	0	1	3	2
7	1	0	1	0	1	1
8	1	0	1	0	1	2
9	1	1	0	1	0	1
10	1	1	0	1	1	1
11	5	0	0	0	1	2
12	5	0	0	0	1	2
13	5	0	1	0	0	1
14	5	0	0	0	0	1
15	2	3	0	1	1	2
16	2	4	1	0	1	1
17	1	1	1	1	1	2
18	2	4	0	1	1	1
19	2	0	1	0	0	1
20	2	2	0	1	2	1
21	2	0	0	0	1	2
22	3	1	0	1	0	1
23	3	3	1	1	1	1
24	3	1	0	1	0	1
25	1	0	1	1	4	1
26	2	1	0	0	3	1
27	3	1	0	0	1	1
28	3	0	0	0	2	1
29	3	0	1	1	1	2
30	3	1	0	1	0	1
31	4	2	0	0	0	1
32	4	1	1	1	0	1
33	4	0	1	0	1	2
34	3	1	0	1	1	1
35	4	0	0	0	1	2
36	4	0	0	0	0	1
37	4	0	1	0	1	1
38	4	0	0	0	1	2
39	5	0	0	0	0	1
40	4	0	0	1	1	2
41	3	0	0	1	1	1
42	3	0	1	0	0	1
43	3	1	1	0	1	1
44	5	0	0	0	1	2
45	4	0	0	1	0	1
46	5	0	0	0	1	2
47	5	0	0	0	0	1
48	5	0	0	0	0	1
49	2	1	0	1	2	1
50	4	1	0	1	0	1

Table 2: Data set for imprecise tree

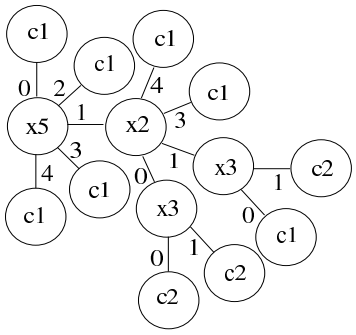

Figure 4: Example 5.1: Tree 2

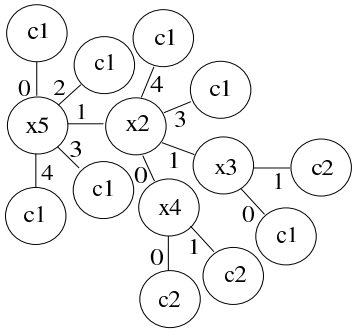

Figure 5: Example 5.1: Tree 3

theoretically. Moreover, the amount of mass given to each unobserved category depends on how the unobserved categories are described, and how many there are, which may cause problems. In contrast, minimising entropy guarantees that no unobserved category will be given any mass, side-stepping the issue of how to label and quantify unobserved categories.

Finally, we should also consider using alternative information measures (one such alternative is the Gini index [10]) to generate imprecise decision trees.

Acknowledgements

This work was supported by the UK National Institute of Health Research, and by the Spanish Consejería de Economía, Innovación y Ciencia de la Junta de Andalucía, under project TIC-06016, which supported the first and second authors respectively. We would like to thank our two reviewers for their comments and suggestions.

References

[1] Abellan, J. (2006) Uncertainty measures on probability intervals from the imprecise Dirichlet model, *International Journal of General Systems*, Vol 35, **5**, 509-528.

[2] Abellan, J. & Moral, S. (2003) Building classification trees using the total uncertainty criterion, *International Journal of Intelligent Systems* 18 (12), 1215-1225.

[3] Abellan, J. & Moral, S. (2005) Difference of entropies as a non-specifity function on credal sets, *Interna-*

tional Journal of General Systems, Vol 34, **3**, 201-214.

[4] Abellan, J. & Masegosa, A. (2010) An ensemble method using credal decision trees, *European Journal of Operations Research*, 205 (1), 218-226.

[5] Abellan, J., Baker, R.M. & Coolen, F.P.A. (2011) Maximising entropy on the nonparametric predictive inference model for multinomial data, *European Journal of Operational Research* , 212(1) 112-122.

[6] Baker, R.M. (2010) *Multinomial Nonparametric Predictive Inference: Selection, Classification and Subcategory Data*, PhD thesis, www.etheses.dur.ac.uk/257/

[7] Coolen, F.P.A., Coolen-Schrijner, P. & Maturi, T.A. (2010) On nonparametric predictive inference for ordinal data. In: E. Hullermeier *et al* (eds), *Computational Intelligence for Knowledge-Based Systems Design*, Springer, pp 188-197.

[8] Hill, B.M. (1968) Posterior distribution of percentiles: Bayes' theorem for sampling from a population, *Journal of the American Statistical Association*, Vol 63, 677-691.

[9] Quinlan, J.R. (1986) Induction of decision trees, *Machine Learning 1*, 81-106.

[10] Strobl, C. & Augustin, T. (2009) Adaptive selection of extra cutpoints - an approach towards reconciling robustness and interpretability in classification trees, *Journal of Statistical Theory and Practice*, Vol 3, 119-135.

[11] Troffaes, M.C.M. (2007) Decision making under uncertainty using imprecise probabilities, *International Journal of Approximate Reasoning*, Vol 45, 17-29.

[12] Walley, P. (1996) Inferences from multinomial data: learning about a bag of marbles, *Journal of the Royal Statistical Society* B 58, 3-57.

[13] Zaffalon, M. (2002) The naive credal classifier. *Journal of Statistical Planning and Inference*, 105 (i1), 5-21.