Neural Network Interest Group

Título/*Title*:
MEE Trees

Autor(es)/*Author(s)*:
J.P. Marques de Sá

Relatório Técnico/*Technical Report* No. 2 /2009

FEUP/INEB, Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Título/*Title*:

MEE Trees

Autor(es)/*Author(s)*:

J.P. Marques de Sá

Relatório Técnico/*Technical Report* No. 2 /2009

Publicado por/*Published by*: NNIG. http://paginas.fe.up.pt/~nnig/

Contents

1 Introduction

Decision tree classifiers are mathematical devices popularly applied to data classification tasks. Proposed in the late sixties as a hierarchical or sequential approach to pattern recognition (see e.g. [10], [30]) they are now widely used in data mining tools. The main advantages of decision trees are the semantic interpretation often assignable to decision rules at each tree node (a relevant aspect e.g. in medical applications), the simplification afforded by dealing at each successive tree node with a simpler problem, and to a certain extent their fast computation (rendering them attractive in data mining applications).

Formally, in classifier design one is given a training set $\mathcal{L} \subset X \times \Omega$, where X is the pattern space whose instances (objects or cases) x_i are feature (predictor) vectors and Ω is the target (class) space whose elements ω_j label in some convenient way the class membership of each instance, $\omega_j = \omega(x_i)$, $j = 1, \ldots, c$; c is the number of classes and ω the class assignment function of X into $\Omega = \{\omega_j\}$. The hierarchical or tree approach to classification, i.e. to a $y : X \rightarrow \Omega$ mapping, uses consecutive domain partitions represented by the application of successive tree node rules until reaching a tree leaf assigning a class label, $y(x_i) \in \Omega$.

Testing a single feature value at each node is by far the most used type of decision rule. This so-called *univariate split* is represented by a binary test y_j in the form of $\{ x_{ij} \leq \Delta,\ y_j(x_i) = \omega_k;\ \overline{\omega}_k,\ \text{otherwise}\}$ for *numerical* features or of $\{ x_{ij} \in B,\ y_j(x_i) = \omega_k;\ \overline{\omega}_k,\ \text{otherwise}\}$ for *categorical* features, with Δ and B representing respectively a numerical threshold and a set of values (categories). Mathematical and implementation details on this sort of binary decision trees can be found in many publications (see e.g. [4, 8, 11, 22, 26). In the present paper we only consider binary trees based on univariate splits, which are by far the most popular trees since they afford a straightforward semantic interpretation and are easily implemented.

There is an abundant literature on tree design approaches aimed at finding an "optimal" tree in the sense of finding a minimum sized tree attaining high classification accuracy (e.g. [2]). Since a search on the whole set of possible trees for a given problem is almost always impractical, strategies based on finding, in a top-down recursive way, *locally* optimal nodes have been employed. There is however a shortcoming with such "optimized" designs, as explained in [15]; whereas node performance is a linear function of the class recognition rates, the total tree performance is a (usually highly) nonlinear function. To cope with this issue more recent approaches of tackling the problem in a *global* bottom-up model-based way were proposed (see e.g. [12]); such approaches seem, however, confined to very simple problems, and where deriving such a model is a feasible task. Thus, finding an "optimal" tree in practical problems remains an elusive goal.

Tree design in a top-down recursive fashion through the optimization of a local, nodal, criterion is usually carried out with algorithms based on the seminal works [4] and [24]. Local criteria popularized by these works are forms of a node disorder-measuring function, known as impurity function. We use from now on the word "node" meaning the subset of $\mathcal{L}$ represented by a node. Since the top-down design is not guaranteed to be "optimal" in any sense, the accuracy, efficiency and generalization capability of the final solution relies on using "reasonable" impurity functions and some sort of tree pruning remedying the usual overfitting to the training set of the derived top-down solution.

In the present paper we describe the application of a new type of node criterion – the entropy of nodal error – to tree design. This criterion, coined MEE (Minimization of Error Entropy) in other works, was successfully applied to the design of other classifier machines, namely artificial neural networks ([28]). The paper is organized as follows: section 2 describes the MEE design approach to trees and its idiosyncrasies; section 3 compares MEE trees with their related competitors both from a theoretical and experimental perspective; section 4 presents in a comparative way the application of MEE trees to real-world datasets; section 5 discusses the pruning issue for MEE trees; finally, the conclusions are drawn in section 6.

2 MEE Trees

2.2 MEE Splits

The main idea of the MEE tree design algorithm consists of viewing a candidate node split (split of the U subset represented by a node u) as a *two-class problem* in $\Omega \times Y$ (set-of-classes×set-of-rules) and determining which $(\omega, y) \in \Omega \times Y$ affords the best solution in the minimum entropy-of-error sense.

Finding the best distribution-free univariate split in a two-class problem is known as a Stoller split problem ([8], [29]). Consider then a candidate split at a node relative to a specific class $\omega_k \in \Omega$ and its complement $\overline{\omega}_k = \cup_{i \neq k} \omega_i$ as a Stoller split problem using a rule $y \in Y$. For any input object $x \in U$ the rule y produces a class assignment $\omega_y(x) \in \{\omega_k, \overline{\omega}_k\}$ which is compared with the true class label $\omega(x)$ as shown in Figure 1. We may further view the determination of the best Δ or B for a given feature variable as if achieved by some kind of adaptive system as also shown in the same figure. Keeping with the terminology of adaptive systems we then have an "error" variable corresponding to $\omega(x) - \omega_y(x)$ (assuming appropriate coding) and an adapting function of the error which is the entropy, $H(error)$.

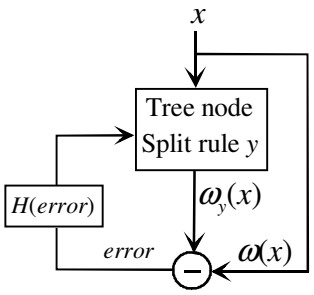

Fig. 1. A node split viewed as an adaptive Stoller split problem.

Let us denote by T and Y the random variables (r.v.) corresponding to a convenient coding of $\omega(x)$ and $\omega_y(x)$, respectively; say, by assigning 1 if $\omega = \omega_k$ and 0 otherwise. We then also have an r.v. of the errors, $E = T - Y$, taking value in {-1, 0, 1}, such that:

$P(E = 1) \equiv P_{10} = P(T = 1, Y = 0)$ is the misclassification probability of class ω_k;

$P(E = -1) \equiv P_{01} = P(T = 0, Y = 1)$ is the misclassification probability of class $\overline{\omega}_k$;

$P(E = 0) = 1 - P_{01} - P_{10}$ is the correct classification probability.

The function of the error "adjusting" the split rule that we use is the Shannon Entropy-of-Error (*EE*):

$$EE = -P_{01} \ln P_{01} - P_{10} \ln P_{10} - (1 - P_{01} - P_{10}) \ln(1 - P_{01} - P_{10}) .$$

The MEE split then consists of finding Δ or B corresponding to min*EE*. (In the computations the 0ln0 = 0 convention applies as usual.)

Note that although *EE* is an impurity function of E it is *not* a node impurity function, such as the Gini index of diversity ([4]) or the node "information" ([24]). Whereas a node impurity function depends solely on the $P(\omega_j \mid \text{node } u)$ probability mass function (pmf), *EE* depends on the "error" of a specific split involving parent and children nodes. Moreover, whereas node impurity functions have a c-dimensional support for any arbitrary c, *EE* always has a 3-point support strictly related to two-class problems.

The entropy-of-error has the following properties:

1. $EE(P_{01}, P_{10})$ is a concave function (surface) on $[0, 1 - P_{10}] \times [0, P_{10}]$, $P_{10} \in [0,1]$ (since it is a sum of concave functions with the same support).

2. $EE(P_{01}, P_{10})$ has three absolute minima (0) at (0,0), (0,1) and (1,0). Note that these three minima correspond to Dirac distributions of E, with the first one being the ideal one for an optimal classifier: no classification errors.
3. $EE(P_{01}, P_{01})$ has one absolute maximum ($ln3$) at $P_{01} = P_{10} = 1/3$.

The motivation for using MEE splits stems from two facts: MEE often outperforms other machine training methods (e.g., the ubiquitous mean square error) in classification, namely in multilayer perceptron training ([28]); MEE will not work for largely overlapped distributions ([27]) and this will provide a natural way where to stop when growing a tree. Regarding this last aspect Figure 2 illustrates the two situations one may encounter. In Figure 2a the overlap of the distributions is small and the minEE point occurs somewhere near the minimum error point. In Figure 2b with large overlap of the distributions the minEE point occurs at an end point of the variable range. In this last case, as explained in [27], the minimum error point tends to occur in the vicinity of an entropy maximum and not of a minimum.

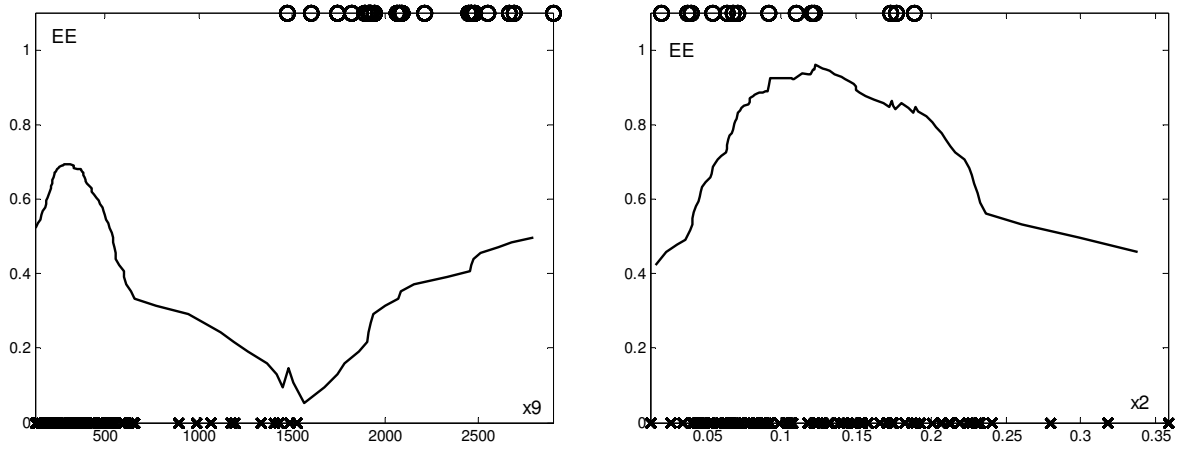

Fig. 2. **Entropy-of-error curves for two splits of the Breast-Tissue dataset (splitting the balls from the crosses): a) feature x_9 with class 6, minEE at x_9=1563.8; b) feature x_2 with class 2, minEE at x_2=0.0162.**

2.3 The MEE Tree Algorithm

The main algorithmic operations for growing a MEE tree are simple enough and resemble those performed when using node impurity measures:

1. At each tree node, $U \subset \mathcal{L}$, we are given an $n \times f$ feature matrix X_U and an $n \times c$ class matrix T_U (filled with zeros and ones).
2. A univariate split y with parameter Δ or B minimizing EE is searched for in the $f \times c$ space.
3. For that purpose, the "error" probabilities are estimated for each candidate class label t as:
$$P_{10} = n_{10}/n, \quad P_{01} = n_{01}/n,$$
with $n_{tt'}$ meaning the number of class t instances classified as t'.
4. The rule minimizing (the empirical) EE is assigned to the node and if a stopping criterion is satisfied the node becomes a leaf. Otherwise, the corresponding left and right node sets are generated and steps 2 and 3 iterated.

In step 2 the middle points of the distinct values of every feature $x_{.i}, i = 1, \ldots, f$ are used as candidate split points Δ_j for numerical-type rules ("$x_{.i} > \Delta_j$?"), and combinations B_j of categorical values for categorical-type rules ("$x_{.i} \in B_j$?").

A leaf is reached whenever a lower bound on the number of instances is reached or min*EE* occurs at interval ends. A previous (and preliminary) version of the algorithm in [19] included also an *EE* threshold as stopping criterion; this was found to be of no use and abandoned. The final class label assignment of the leaves is made by the usual majority voting rule (it will differ from the MEE candidate class only in very rare cases).

The following three issues are characteristic of these trees.

2.3.1 Candidate class position for numerical variables

The entropy-of-error for numerical variables has to be computed for two distinct class configurations: the candidate class ($t = 1$) corresponds to the $x_{.i} > \Delta_j$ rule; the candidate class ($t = 1$) corresponds to the $x_{.i} \le \Delta_j$ rule. The reason for this is twofold: the class position is not known *a priori*; *EE* is dependent on the class position (see below).

Fig. 3 The two distinct configurations of the candidate class (black meaning *t*=1 and white *t*=0; blacks and whites are as continuous strips in the figure for ease of representation only).

It is, however, a trivial task to compute the *EE* for one of the configurations once the numbers of errors are computed for the other. Let us assume that the computation for the left configuration of Figure 3 has been carried out. In pseudo code:

$rule \leftarrow (x > delta)$

$n_{10} \leftarrow sum(\text{not } rule \text{ and } t)$

$n_{01} \leftarrow sum(rule \text{ and not } t)$

Using these *n*'s one now computes their values for the other configuration without having to carry out the application of the decision rule: $n_1 \leftarrow sum(t)$; $n_{10} \leftarrow n_1 - n_{10}$; $n_{01} \leftarrow n - n_1 - n_{01}$.

The class position issue does not apply to categorical variables (no order relation).

2.3.2 Handling categorical variables

Tree growing based on concave impurity functions enjoys a very interesting feature when handling categorical variables of a set $B = \{b_1, \cdots, b_m\}$: one does not have to analyze all $2^m - 2$ rules $x_{.i} \in B_j = \{b_{j1}, \ldots, b_{jk}\}, b_{ji} \in B$. One only has to analyze *m* subsets B_j ([4]). No similar result exists for MEE trees. However, the computation of the $2^m - 2$ rules has only to be fully carried out for the *m* categories. This represents a considerable time saving. We now prove this. Denoting a singleton category set $\{b\}$ by *b* and its complement, $B-\{b\}$, by $\bar{b}$, we have:

		Category b	Category $\bar{b}$
Class	ω_1	$P(1,b)$	$P(1,\bar{b})$
	ω_0	$P(0,b)$	$P(0,\bar{b})$

The pmf of *E* has three values:

$$P_{10} = P(T = 1, Y = 0) = P(1,\bar{b}) \equiv P(\omega_1, x \notin \{b\});$$

$$P_{01} = P(T=0, Y=1) = P(0,b) \equiv P(\omega_0, x \in \{b\});$$
$$P_0 = 1 - P(1,\bar{b}) - P(0,b) = P(1,b) + P(0,\bar{b}).$$

We have already seen that we only need to compute the two values P_{10} and P_{01} in order to compute *EE*. Denoting by *p* and *q* the class priors we have:

$$P(\bar{b} \mid 1) = P_{10} / p; \;\; P(b \mid 1) = 1 - P_{10} / p; \;\; P(b \mid 0) = P_{01} / q.$$

We now consider non-singleton sets, B_i. We first notice that:

$$P(\{B_1, B_2\} \mid \omega) = P(B_1 \mid \omega) + P(B_2 \mid \omega) \text{ for } B_1 \cap B_2 = \varnothing.$$

Therefore, once the two P_{10} and P_{01} values have been computed for all *m* singleton sets it is an easy task to compute them for any non-singleton category set *B*, since:

$$P(B \mid 1) = \sum_{b \in B} P(b \mid 1); \;\; P(B \mid 0) = \sum_{b \in B} P(b \mid 0).$$

Thus: $P_{10} = p(1 - P(B \mid 1)); \quad P_{01} = qP(B \mid 0)$.

These results hold for both the theoretical and empirical pmf's.

2.3.3 Entropy-of-Error of Class Unions

Since the MEE approach is essentially a two-class discrimination approach, one may expect to obtain performance improvements for datasets with $c > 3$ classes by considering class unions, i.e., by including unions of classes, say of *k* classes with *k* up to [*c*/2] (greatest integer less than *c*/2) in the set of candidate classes. The difficulty thereof is that the number of candidate classes may become quite high with an accordingly higher computation time. For instance, for a 6 class problem one would have consider the $\binom{6}{2} = 15$ unions of 2 classes and $\binom{6}{3} = 20$ unions of 3 classes, amounting to a total of 41 classes. There is, however, a fast way of computing the errors for unions of classes as we shall now show. We first notice that:

$$P_{10} \equiv P(T=1, Y=0) = \frac{n_1}{n}\frac{n_{10}}{n_1} = \frac{n_{10}}{n},$$
$$P_{01} \equiv P(T=0, Y=1) = \frac{n_0}{n}\frac{n_{01}}{n_0} = \frac{n_{01}}{n},$$
$$P_0 \equiv P((T=0, y=0) \bigcup (T=1, Y=1)) = 1 - P_{10} - P_{01},$$

where we assigned the candidate class, ω_1, to the node that satisfies the rule, ω_0 is its complement, and named the *n*'s accordingly.

Consider the class union $\omega = \omega_1 \cup \omega_2$, $\omega_1 \cap \omega_2 = \varnothing$, and suppose that we have computed for ω_i, $i = 1, 2$, the following three quantities:

1. $n_{10}(\omega_i)$ – number of instances of class *i* that do not satisfy the rule;
2. $n_{11}(\omega_i)$ – number of instances of class *i* that satisfy the rule;
3. n_r – number of instances (from whatever class) that satisfy the rule *r*.

Quantity 3 is computed only once for each feature. Quantities $n_{10}(\omega_i)$ and $n_{01}(\omega_i)$ have been used to compute *EE*. For the reasons shown below we compute $n_{11}(\omega_i)$ instead of $n_{01}(\omega_i)$. We have:

$$P_{10}(\omega_i) = P(\bar{r}, \omega_i) = P(\bar{r} \mid \omega_i)P(\omega_i) = \frac{n_{10}(\omega_i)}{n_1(\omega_i)}\frac{n_1(\omega_i)}{n} = \frac{n_{10}(\omega_i)}{n};$$

$$P(r,\omega_i) = P(r \mid \omega_i)P(\omega_i) = \frac{n_{11}(\omega_i)}{n_1(\omega_i)}\frac{n_1(\omega_i)}{n} = \frac{n_{11}(\omega_i)}{n};$$

$$P(r,\overline{\omega}_i) + P(r,\omega_i) = P(r) \quad \Rightarrow \quad P_{01}(\omega_i) = P(r) - P(r,\omega_i) = \frac{n_r}{n} - \frac{n_{11}(\omega_i)}{n}.$$

With n_{10} and n_{11} we are thus able to compute the *EE* for the original classes. Moreover:

$$P_{10}(\omega) = P(\overline{r},\omega) = P(\overline{r},\omega_1) + P(\overline{r},\omega_2) = P_{10}(\omega_1) + P_{10}(\omega_2);$$

$$P_{01}(\omega) = P(r,\overline{\omega}) = P(r) - P(r,\omega) = P(r) - \left[P(r,\omega_1) + P(r,\omega_2)\right] =$$
$$= P_{01}(\omega_1) + P_{01}(\omega_2) - P(r).$$

It is therefore a negligible time-consuming task to compute the probabilities of interest for the union of classes without having to explicitly carry out the corresponding decision test. These results hold for both the theoretical and empirical pmf's.

3 Entropy-of-Error and Impurity Functions

3.1 Impurity Functions

An impurity function of a node u, $\phi(u) \equiv \phi\left(P(\omega_1 \mid u),\ldots,P(\omega_c \mid u)\right)$, measures the degree of "disorder" of its pmf in such a way that: a) ϕ achieves its maximum at (1/c, 1/c,…, 1/c) (uniform distribution); b) ϕ achieves its minimum at (1,0,…,0), (0,1,…,0),…,(0,0,…,1) (Dirac distribution); c) ϕ is symmetric. It is a simple generalization of the entropy notion. Celebrated impurity functions in the bibliography and applications are the Gini diversity index, $\phi(u) \doteq GDI(u) = \sum_{j=1}^{c}\sum_{k=1,k\neq j}^{c} P(\omega_j \mid u)P(\omega_k \mid u) = 1 - \sum_{j=1}^{c} P^2(\omega_j \mid u)$ and the Shannon entropy (also called "information") $\phi(u) \doteq H(u) = -\sum_{k=1}^{c} P(\omega_k \mid u)\ln P(\omega_k \mid u)$. Other definitions of entropy such as the Rényi or Tsallis entropies have also been used ([20]). The Gini diversity index can be interpreted as a variance measure ([25]).

Consider a node u, candidate to splitting into right and left nodes, u_r and u_l. The average impurity is $\phi_y(u) = P(u_r \mid u, y)\phi(u_r) + P(u_l \mid u, y)\phi(u_l)$, called Gini index (*GI*) and information gain (*IG*), respectively for *GDI*(u) and $H(u)$, in [4] and [22-24]. One may then compute and maximize $\Delta_y\phi(u) = \phi(u) - \phi_y(u)$ at each node (contributing to a higher decrease of impurity). Since $\phi(u)$ doesn't depend on the rule one may equivalently minimize $\phi_y(u)$. This is what we will do below.

One may also consider the probability of error (*PE*) at a node as an impurity function, $\phi(u) \doteq PE(u) = 1 - \max_j P(\omega_j \mid u)$, since it enjoys all the above properties. However, as shown in [4], for tree design $PE(u)$ has two serious defects: it may lead to $\omega(u) = \omega(u_r) = \omega(u_l)$ for all candidate splits when one of the classes has a predominant number of instances (and the majority rule is used) and in such cases no best split can be found; it may lead to reject solutions with purer children nodes in detriment of others with highly impure nodes resulting in worse performing trees. For two-class splits with p and $q = 1 - p$ being the respective class probabilities, $GI(p) = 2pq$ and $IG(p) = -p\ln p - q\ln q$ are both concave functions whereas $PE(p) = \min(p,q)$ P_e is not strictly concave, justifying as shown in [4] the insufficiencies of *PE*.

3.2 Comparing IG, PE and EE

In this section we compare the *IG*, *PE* and *EE* criteria. The reason for electing to compare *EE* with *IG* and *PE* and not other criteria (including those that do not correspond to impurity functions) has to do with the fact that, as shown in the continuation, *EE* has a kind of intermediate behavior between *IG* and *PE*. Also, since *GDI*(u) is simply the first order

approximation of $H(u)$, both *GI* and *IG* measures behave in a similar way (a 2% disagreement between *GI* and *IG* is reported in [25]) allowing us to focus solely on *IG*.

3.2.1 Theoretical analysis

We carry out the comparison in the following framework: we consider two classes, ω_1 and $\omega_0 \equiv \overline{\omega}_1$, and the splitting of a node with n_1 instances from class 1 and n_0 from class 0 as shown in Figure 4.

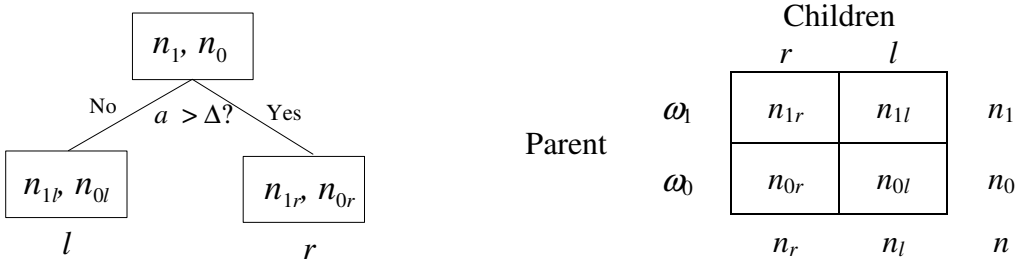

Fig. 4. Two-class framework.

At the parent node, with $n = n_1 + n_0$ instances, we have (resubstitution estimates throughout): $P(\omega_1) = n_1 / n = p$; $P(\omega_0) = n_0 / n = 1 - p = q$. In order to compare the several splitting criteria we consider that for any prior $p = P(\omega_1)$ and whatever criterion being used a certain percentage α of ω_1 instances are assigned to the right node and a certain percentage β of ω_0 instances are assigned to the left node. In other words, assuming that the labels assigned to the right and left nodes are respectively ω_1 and ω_0, α and β are the percentages of instances, respectively from classes 1 and 0, that are correctly classified: $n_{1r} = \alpha n_1$; $n_{0l} = \beta n_0$.

Denoting $n_r = n_{1r} + n_{0r}$, $n_l = n_{1l} + n_{0l}$, we have:

Information-Gain criterion

Entropy of the right node: $$H_r = -\frac{n_{1r}}{n_r}\ln\frac{n_{1r}}{n_r} - \frac{n_{0r}}{n_r}\ln\frac{n_{0r}}{n_r}$$

Entropy of the left node: $$H_l = -\frac{n_{1l}}{n_l}\ln\frac{n_{1l}}{n_l} - \frac{n_{0l}}{n_l}\ln\frac{n_{0l}}{n_l}$$

Average entropy to be minimized: $IG(p,\alpha,\beta) = \frac{n_r}{n}H_r + \frac{n_l}{n}H_l =$

$$-\alpha p\ln\frac{\alpha p}{\alpha p + (1-\beta)q} - (1-\beta)q\ln\frac{(1-\beta)q}{\alpha p + (1-\beta)q} - (1-\alpha)p\ln\frac{(1-\alpha)p}{(1-\alpha)p + \beta q} - \beta q\ln\frac{\beta q}{(1-\alpha)p + \beta q}$$

Note that $IG(p,\alpha,\beta)$ (as well as $GI(p,\alpha,\beta)$) is *independent* of the predicted classification. That is, swapping α by β together with p by q in the above formula leads to the same result. In contrast the following two criteria are *not independent* of the predicted classification (and the last one is not a node impurity criterion).

Probability-of-Error criterion:

Assuming the assignment $r = \omega_1$, $l = \omega_0$ and denoting by PE_1, PE_0 the class error rates:

$$PE_1 = \frac{n_{1l}}{n_1} = 1 - \alpha;\ PE_0 = \frac{n_{0r}}{n_0} = 1 - \beta;$$

$$PE = pPE_1 + qPE_0 = p(1-\alpha) + q(1-\beta) = \frac{n_{1l} + n_{0r}}{n}$$

Had we swapped the assigned classes we would obtain in general a different value:

$$PE = pPE_1 + qPE_0 = p\alpha + q\beta = \frac{n_{1r} + n_{0l}}{n}$$

Entropy-of-Error criterion:

$$P_{10} \equiv P(E=1) = p(1-\alpha) = \frac{n_1}{n}\frac{n_{1l}}{n_1} = \frac{n_{1l}}{n};$$

$$P_{01} \equiv P(E=-1) = q(1-\beta) = \frac{n_0}{n}\frac{n_{0r}}{n_0} = \frac{n_{0r}}{n};$$

$$P_0 \equiv P(E=0) = p\frac{n_{1r}}{n_1} + q\frac{n_{0l}}{n_0} = \frac{n_{1r} + n_{0l}}{n} = \alpha p + \beta q = 1 - P(E=1) - P(E=-1)$$

To these values we apply: $EE = -P_{10}\ln P_{10} - P_{01}\ln P_{01} - P_0 \ln P_0$. Again, for swapped assignments one obtains in general a different value.

We now analyze three categories of configurations:

Configurations with p = 0

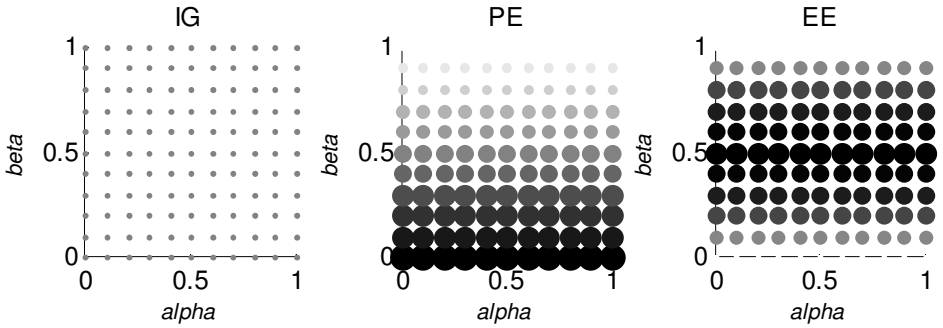

Fig. 5. The p = 0 setting. Darker colors and larger sizes correspond to higher values.

For p = 0 there are no class 1 instances and the value of α is immaterial. As illustrated in Figure 5, *IG* (and *GI*) is always zero no matter how β is chosen because the descendent nodes are always pure. *PE* grows linearly with decreasing β, because for smaller β larger numbers of class 0 instances are sent to the wrong side. *EE* is concave and maximum at $\beta = 0.5$ and minimum for $\beta = 0$, 1. Figure 6 shows error pmf's for three different values of β. Notice that for $\beta = 0$ all instances would be wrongly classified; however, this doesn't happen because of the majority rule. *IG* (and *GI*) are unable to discriminate these configurations.

Similar conclusions are extracted for the p = 1 configuration.

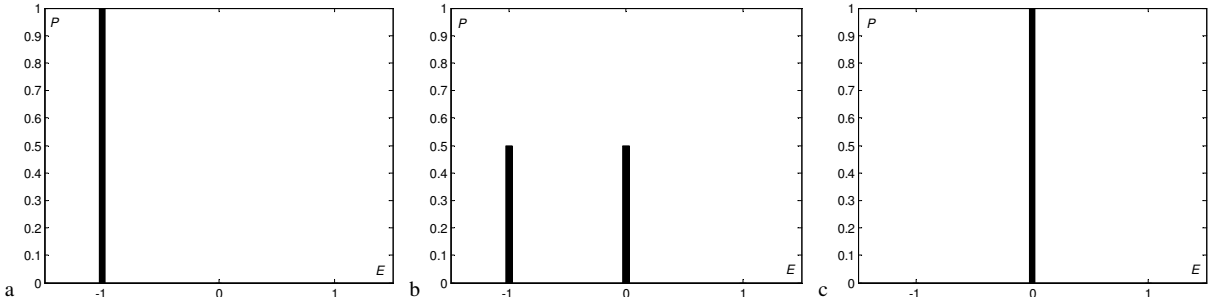

Fig. 6. Error pmf's for p = 0 and three different values of β: a) β = 0 (*EE* = 0); b) β = 0.5 (*EE* = 0.69); c) β = 1 (*EE* = 0) .

Configurations with p = 0.5

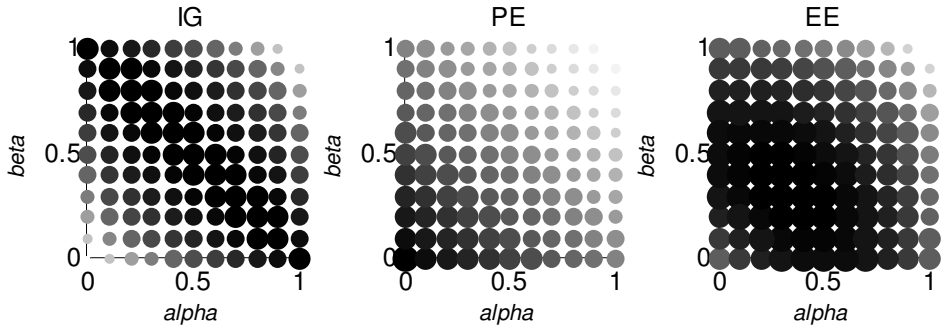

Fig. 7. The p = 0.5 setting. Darker colors and larger sizes correspond to higher values.

As shown in Figure 7 *IG* and *EE* are concave functions; *PE* is a linear function. Figure 8 shows the error pmf's for three combinations of α, β. Note that *IG* found cases 'a' and 'b' to be similar, which seems rather inadequate, whereas both *PE* and *EE* found 'b' to be an improvement over 'a'. Cases 'b' and 'c' correspond to those analyzed in [4] to illustrate why *PE* should not be used. Note that *IG* selects 'c' with a decrease of 16%; *EE* also selects 'c' and with a more pronounced decrease: 22%. Therefore, for equal probability of error *EE* also prefers purer nodes. In all cases *GI* behaves similarly to *IG*.

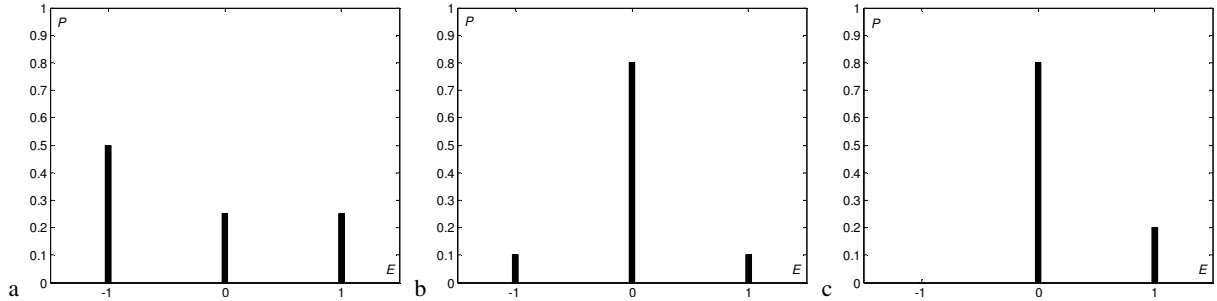

Fig. 8. Error pmf's for p = 0.5: a) α = 0.5, β = 0 (*IG*=0.48, *PE*=0.75, *EE*=1.04); b) α = 0.8, β = 0.8 (*IG*=0.50, *PE*=0.20, *EE*=0.64); c) α = 0.6, β = 1 (*IG*=0.42, *PE*=0.20, *EE*=0.50).

Configurations with $\alpha = \beta$

For these configurations *PE* doesn't depend on p ($PE = 1-\alpha$). It is then possible to show the behavior in terms of p and *PE* as in Figure 9. We observe that *EE* is concave, symmetric in p but asymmetric in *PE*! In conclusion, *EE* displays, just as *GI* and *IG*, a concave behavior but sensitive to the *PE* value. Moreover, for fixed p, $EE(p)$ only starts from zero for $PE = 0$ or $PE = 1$; otherwise, the initial deviation from zero depends on the *PE* value.

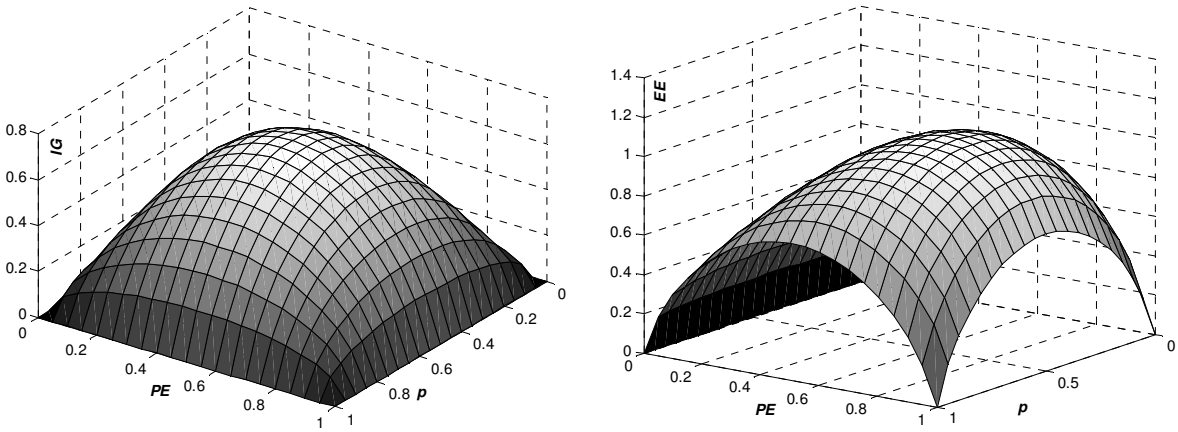

Fig. 9. *IG* and *EE* for $\alpha = \beta$.

3.2.2 Experiments with normal distributions

We experimentally studied the convergence of the (empirical) error rates produced by the *IG* and *EE* rules applied to Stoller split problems with normal distributions. In this setting the theoretically optimal error rates are known. The experiments controlled the studentized difference of the class means, $d = (m_1 - m_2)/s_p$ (s_p is the pooled standard deviation), as well as the number of class instances (equal for both classes), n, which we chose to vary from 10 to 200. We also computed the frequency of interval-end hits in all experiments.

For a fixed d and each value of n we performed 100 experiments obtaining in this way reliable estimates of the empirical mean error. The following conclusions could then be drawn from a large number of experiments with different d values: for a not too small d (say, $d >$ 0.5) the empirical error rate produced by the *IG* rule converges to the optimal error rate and only for low values of n ($n < 20$) there is a noticeable frequency of interval-end hits; in the previous conditions the error rate of the *EE* rule does not converge to the optimal error rate (it is slightly higher) and the frequency of interval-end hits remains significant even for high values of n; only for higher values of d (say, $d > 1.5$) a convergence of the error rate of the *EE* rule to the optimal value is observed and the frequency of interval-end hits is practically zero.

As a matter of fact, by appropriate experiments covering $d \in [0.6, 2.2]$ we could locate the turnover point between convergence and no-convergence to the optimal error rate at around d=1.4, in agreement with the theoretical findings published in [27]. Below this "no-convergence value" *EE* exhibits invalid solutions with split points at interval ends. The stop-splitting rule mentioned in 2.3 (min*EE* occurs at interval ends) is therefore justified.

4 Application to Real-World Datasets

4.2 Experimental Setup

The MEE algorithm was implemented in Matlab (version 7.2)[1], applied to the 24 datasets presented in Table 1, and the results confronted with those obtained using the CART-Gini, CART-Information-Gain (also known as "CART-deviance") and CART-Twoing algorithms available in Matlab and the C4.5 algorithm available in Weka. The "towing" split rule corresponds to maximizing the variance of class conditioned right and left pmf estimates ([4]); it is not related to an impurity function.

As shown in Table 1 the datasets are quite diverse in terms of number of instances, features and classes as well as of feature types. The following ones are from the well-known UCI repository, [3]: balance, car (evaluation), Cleveland heart disease (2 classes), dermatology, E-coli, glass, image segmentation, ionosphere, led (display), page blocks, Pima (indian) diabetes, (E-coli) promoter gene (sequences), (New York) thyroid (database), Wisconsin diagnostic breast cancer, yeast and zoo. We removed classes "omL", "imL" and "imS" from the E-coli dataset because of their low number of instances (respectively, 5, 2, 2). For the same reason we removed class "erl" from yeast (5 instances). We disregarded the 6 instances (out of the original 303) of the Cleveland HD2 dataset that have missing data. We also disregarded 2 features (out of 8) of the yeast dataset which clearly have no discriminative value (constant value in all or almost all instances).

[1] The Matlab implementation of the MEE algorithm is available at http://paginas.fe.up.pt/~nnig/.

Table 1. Datasets. The number of categorical features is given inside parentheses.

	Balance	Breast	Breast4	Car	Clev. HD2	Colon
No. cases	625	106	106	1278	297	62
No. features	4 (4)	9 (0)	9 (0)	6 (6)	13 (8)	2000 (0)
No. classes	3	6	4	4	2	2
	CTG	Dermatol.	E-coli	E-coli4	Glass	Image Seg.
No. cases	2126	358	327	327	214	2310
No. features	21 (0)	34 (33)	5 (0)	5 (0)	9 (0)	18 (0)
No. classes	10	6	5	4	6	7
	Ionosphere	Led	Leukemia	Page blks	Pb12	P. Diabetes
No. cases	351	200	72	5473	608	768
No. features	34 (0)	7 (7)	7129 (0)	10 (0)	2 (0)	8 (0)
No. classes	2	10	2	5	4	2
	P. Gene	Spect-Heart	Thyroid	Wdbc	Yeast	Zoo
No. cases	106	267	215	569	1479	101
No. features	57 (57)	22 (22)	5 (0)	30 (0)	6 (0)	16 (16)
No. classes	2	2	3	2	9	7

The colon and leukemia datasets are from the Kent Ridge Biomedical Dataset (http://datam.i2r.a-star.edu.sg/datasets/krbd); they are described in detail respectively in [1] and [13]. The pb12 dataset is from [14]. The breast (tissue) and CTG datasets are available and described in [18]. As shown in this work classes "fad", "mas" and "gla" of the breast dataset have a large overlap and cannot be discriminated in any reasonable way; this led us to merge them and set up the breast4 dataset. The same reason led us to set up the E-coli4 dataset by merging classes "im" and "imU" of E-coli.

All algorithms were run with unit misclassification costs (tree costs are error rates), estimated priors and the same minimum number of instances for a node to be split: 5. The CART and MEE algorithms were run with Cost-Complexity Pruning (CCP) with the 'min' criterion and 10-fold cross-validation ([4]). The C4.5 algorithm was run with Pessimistic Error Pruning (PEP) at 25% confidence level ([9]).

Besides of resubstitution error estimates we also applied the leave-one-out (LOO) cross-validation procedure which is known to be asymptotically unbiased and converging from above towards the theoretically optimal error rate. Confusion matrices and estimates of the probability of error were computed as well as statistics regarding the tree sizes (number of nodes).

4.3 Results

All results obtained for the five methods (CART-Gini, CART-Information-Gain and CART-Twoing algorithms are from now on simply denoted Gini, Info Gain and Twoing, respectively) were compared as recommended in [7], namely using the nonparametric Friedman test and when a significant $p < 0.05$ was found with a post-hoc Bonferroni multiple comparison test.

4.3.1 Error Rates

Table 2 presents the LOO estimates of the error rate, P_e. For $c > 3$ Table 2 lists the best MEE solution that was found for class unions up to $[c/2]$ (see section 2.3.3). The Friedman test did not detect significant differences ($p = 0.628$) for these 24 datasets. The mean ranks for the five methods (following from now on the Table 2 order) are: 2.875, 2.688, 2.917, 3.313 and 3.208.

Table 2. LOO estimates of P_e.

	Balance	Breast	Breast4	Car	Clev. HD2	Colon
Gini	0.1955 (0.016)	0.3610 (0.047)	0.1892 (0.038)	0.0422 (0.006)	0.2121 (0.024)	0.2258 (0.053)
Info Gain	0.2848 (0.018)	0.2925 (0.044)	0.0849 (0.027)	0.0417 (0.006)	0.2256 (0.024)	0.1935 (0.050)
Twoing	0.2208 (0.017)	0.3585 (0.047)	0.0849 (0.027)	0.0446 (0.006)	0.2290 (0.024)	0.2258 (0.053)
C4.5	0.2256 (0.017)	0.4151 (0.048)	0.1415 (0.034)	0.0475 (0.006)	0.1919 (0.023)	0.2581 (0.056)
MEE	0.3056 (0.018)	0.4340 (0.048)	0.1321 (0.033)	0.0764 (0.007)	0.2222 (0.024)	0.1613 (0.047)
	CTG	Dermatol.	E-coli	E-coli4	Glass	Image Seg.
Gini	0.1604 (0.008)	0.0587 (0.012)	0.1713 (0.021)	0.0948 (0.016)	0.3178 (0.032)	0.0437 (0.004)
Info Gain	0.1811 (0.008)	0.0587 (0.012)	0.1896 (0.022)	0.1040 (0.017)	0.3458 (0.033)	0.0273 (0.003)
Twoing	0.1712 (0.008)	0.0698 (0.013)	0.1988 (0.022)	0.0979 (0.016)	0.2944 (0.032)	0.0372 (0.004)
C4.5	0.1750 (0.008)	0.0754 (0.014)	0.2110 (0.023)	0.0979 (0.016)	0.3458 (0.033)	0.0363 (0.004)
MEE	0.1839 (0.008)	0.0726 (0.014)	0.1407 (0.019)	0.0856 (0.015)	0.3084 (0.032)	0.0584 (0.005)
	Ionosphere	Led	Leukemia	Page blks	Pb12	P. Diabetes
Gini	0.1140 (0.017)	0.3600 (0.034)	0.1944 (0.047)	0.0322 (0.002)	0.0888 (0.012)	0.2370 (0.015)
Info Gain	0.1054 (0.016)	0.0315 (0.033)	0.1944 (0.047)	0.0338 (0.002)	0.0921 (0.012)	0.2331 (0.015)
Twoing	0.1111 (0.017)	0.3150 (0.033)	0.2361 (0.050)	0.0311 (0.002)	0.0921 (0.012)	0.2331 (0.015)
C4.5	0.1339 (0.018)	0.3450 (0.034)	0.2500 (0.051)	0.0322 (0.002)	0.0954 (0.012)	0.2617 (0.016)
MEE	0.1311 (0.018)	0.3100 (0.033)	0.2222 (0.049)	0.0360 (0.003)	0.1447 (0.014)	0.2760 (0.016)
	P. Gene	Spect-Heart	Thyroid	Wdbc	Yeast	Zoo
Gini	0.4245 (0.048)	0.2210 (0.025)	0.0791 (0.018)	0.0756 (0.011)	0.4104 (0.013)	0.1188 (0.032)
Info Gain	0.3019 (0.045)	0.1723 (0.023)	0.0791 (0.018)	0.0685 (0.011)	0.4151 (0.013)	0.1188 (0.032)
Twoing	0.3868 (0.047)	0.2135 (0.025)	0.0837 (0.019)	0.0733 (0.011)	0.4023 (0.013)	0.1287 (0.033)
C4.5	0.2075 (0.034)	0.1648 (0.023)	0.0465 (0.014)	0.0562 (0.010)	0.4462 (0.013)	0.0792 (0.027)
MEE	0.1698 (0.036)	0.1436 (0.021)	0.0791 (0.018)	0.0668 (0.010)	0.5382 (0.013)	0.0990 (0.030)

Reference [5] reports a superiority of CCP (min) over other pruning methods, including PEP. Reference [9] also confirms the good performance of CCP and shows that the PEP method has a tendency to underprune. Based on these findings one may suspect that the C4.5 error rates may be slightly optimistic compared to those found by the other methods.

Besides of the LOO estimates of P_e we also computed the respective resubstitution estimates. We were therefore able to appreciate the generalization ability of the three methods in the 24 datasets. Denoting by e_R and e_{LOO} respectively the resubstitution and LOO error rates, and their standard deviations by s_R and s_{LOO}, we have computed the ratio $D = |e_R - e_{LOO}| / \bar{s}$ using the pooled standard deviation $\bar{s} = \sqrt{(s_r^2 + s_{LOO}^2)/2}$. D reflects the generalization ability of the classifiers, since it reflects the studentized spread between the optimistic e_R (on average terms) and the pessimistic e_{LOO} (on average terms).

The Friedman test found a significant difference (p=0.004) of the methods for the D values (mean ranks: 3.000, 2.521, 3.438, 3.792 and 2.250), the post-hoc test revealing a significant difference between MEE and C4.5 (and also Twoing if we relax the test significance to p=0.09). A better generalization of MEE relative to C4.5 in the 24 datasets seems therefore evident (see Figure 10).

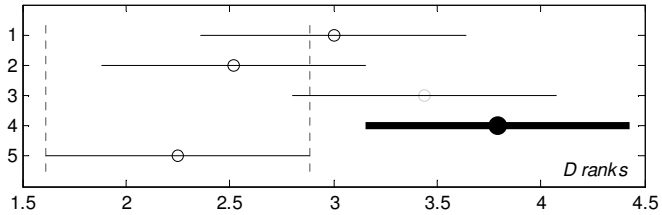

Fig. 10. Multiple comparison Bonferroni post-hoc test for the D ranks (methods are numbered as: 1=Gini; 2=Info Gain; 3=Twoing; 4=C4.5; 5=MEE).

4.3.2 Tree Sizes

Table 3 presents the computed tree size statistics in the LOO experiments. In what concerns average tree size the MEE algorithm achieved the smallest value 14 times; in contrast the

other algorithms produced the smallest values 8, 2, 3 and 1 time, respectively for Gini, Info Gain, Twoing and C4.5. The Friedman test found a significant difference ($p \approx 0$) with column mean ranks (2.583, 3.042, 2.854, 2.208 and 4.313) also hinting at smaller tree sizes for MEE. The post-hoc comparison test found that C4.5 produced sizes significantly different (larger).

Table 3. Mean (standard deviation; range) of the tree sizes in the leave-one-out experiments.

	Balance	Breast	Breast4	Car	Clev. HD2	Colon
Gini	26.6 (7.8; 100)	9.8 (2.0; 10)	7.3 (1.1; 6)	62.1 (7.2; 34)	9.0 (2.2; 8)	3.1 (0.6; 4)
Info Gain	33.5 (9.8; 108)	9.9 (1.0; 12)	7.5 (1.4; 6)	62.6 (9.8; 50)	9.0 (2.3; 12)	3.4 (1.2; 6)
Twoing	30.0 (8.1; 102)	10.3 (2.8; 14)	7.8 (1.7; 6)	63.0 (8.5;42)	9.0 (2.3; 12)	3.1 (0.7; 6)
C4.5	45.3 (4.1; 12)	15.2 (1.0; 8)	10.8 (0.9; 4)	71.0 (0.3; 8)	19.1 (1.4; 10)	6.9 (0.5; 2)
MEE	118.5 (23.4; 122)	10.3 (1.3; 6)	7.0 (0.0; 0)	106.9 (12.1; 56)	26.1 (11.7; 46)	3.2 (0.6; 2)
	CTG	Dermatol.	E-coli	E-coli4	Glass	Image Seg.
Gini	70.6 (11.2; 82)	13.2 (0.6; 6)	10.5 (2.2; 14)	7.8 (1.7; 8)	14.3 (3.5; 22)	88.9 (17.1; 86)
Info Gain	77.6 (19.3; 84)	16.8 (1.1; 6)	10.6 (3.3; 16)	7.8 (1.4; 8)	17.2 (4.1; 24)	66.6 (8.7; 54)
Twoing	72.6 (12.3; 62)	15.9 (1.2; 6)	10.3 (3.1; 16)	7.6 (1.4; 8)	16.2 (3.9; 26)	77.3 (17.7; 86)
C4.5	158.9 (1.4; 22)	13.0 (1.4; 2)	20.5 (4.4; 14)	9.1 (0.3; 2)	35.6 (2.6; 16)	63.0 (0.6; 16)
MEE	65.5 (3.0; 28)	11.1 (0.3; 2)	9.0 (0.3; 2)	7.0 (0.3; 2)	16.9 (1.9; 10)	30.2 (1.8; 6)
	Ionosphere	Led	Leukemia	Page blks	Pb12	P. Diabetes
Gini	5.2 (1.5; 18)	22.2 (5.3; 24)	3.1 (0.3; 2)	24.2 (8.4; 42)	30.9 (6.0; 28)	7.0 (6.4; 36)
Info Gain	5.2 (1.4; 12)	22.9 (6.3; 26)	3.0 (0.2; 2)	24.2 (8.8; 36)	34.4 (5.8; 28)	7.0 (6.2; 42)
Twoing	5.1 (0.6; 8)	23.8 (5.9; 22)	3.0 (0.2; 2)	25.7 (7.9; 34)	30.7 (5.9; 26)	7.2 (6.7; 42)
C4.5	18.6 (3.1; 14)	22.8 (0.9; 6)	4.7 (0.7; 2)	51.0 (0.4; 12)	33.0 (0.7; 10)	24.7 (1.6; 16)
MEE	9.9 (1.1; 4)	22.8 (1.2; 4)	3.0 (0.0; 0)	21.3 (1.3; 8)	17.0 (0.3; 4)	3.0(0.1; 4)
	P. Gene	Spect-Heart	Thyroid	Wdbc	Yeast	Zoo
Gini	10.0 (3.3; 14)	11.9 (8.1; 46)	9.3 (3.1; 12)	10.0 (2.1; 12)	21.9 (3.3; 26)	11.1 (0.6; 6)
Info Gain	7.9 (3.3; 14)	13.7 (7.8; 36)	8.5 (1.8; 12)	10.9 (3.5; 16)	24.2 (5.2; 44)	10.9 (0.4; 2)
Twoing	9.9 (3.6; 16)	11.8 (7.7; 42)	9.3 (3.2; 14)	9.9 (2.1; 14)	24.6 (6.5; 34)	11.0 (0.7; 8)
C4.5	11.3 (1.5; 8)	14.9 (0.4; 2)	9.0 (0.3; 4)	15.3 (1.3; 8)	154.3 (5.3; 54)	11.0 (0.0; 0)
MEE	7.8 (1.5; 6)	3.0 (0.0; 0)	5.0 (0.2; 2)	6.2 (1.0; 2)	27.3 (1.8; 12)	11.0 (0.0; 0)

Using Table 3 results one may also compute an "average node efficiency" – representing how efficient a node is on average in contribution to the correct classification rate –, as $N = (1 - P_e)/\overline{T}$, where $\overline{T}$ is the average tree size. The Friedman test found a significant difference ($p \approx 0$) for the N scores, with column mean ranks (3.375, 3.000, 3.000, 1.708 and 3.917) hinting at MEE being more efficient. As shown in Figure 11a the post-hoc comparison test found C4.5 as being significantly less efficient.

Finally, we analyzed the tree size ranges, R = max(tree size) – min(tree size). A significant difference ($p \approx 0$) was found by the Friedman test with column mean ranks (3.458, 3.958, 3.875, 2.083 and 1.625) hinting at MEE being more stable. The post-hoc comparison test revealed, as shown in Figure 10b, the MEE and C4.5 designs as being significantly more stable ([16]).

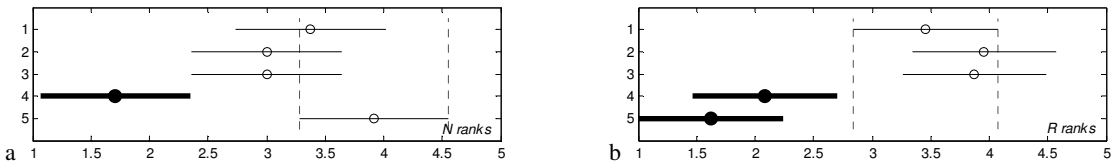

Fig. 11. Multiple comparison Bonferroni post-hoc test (methods are numbered as: 1=Gini; 2=Info Gain; 3=Twoing; 4=C4.5; 5=MEE): a) Tree efficiency; b) Tree size ranges.

5 The Pruning Issue

Tree pruning is a means of dealing with overfitted tree designs that are too biased by the training set. By removing subtrees at deep levels simpler trees are obtained with better generalization capabilities. Several authors (notably [4] and [21]) have expressed the idea that the split criteria are not very influential on the final tree solution and that one should instead concentrate on the pruning operation. In that line of thought even a random rule selection has been proposed and argued that when used with pruning it does not make much difference from other elaborate splitting criteria ([21]). However, one should consider such statements with caution. As a matter of fact, it is shown in [5-6, 17] that there may exist large differences of performance depending on the splitting rule (with a particular bad behavior of the random splitting rule). In [17] it is again confirmed that the choice of splitting rule can be important for certain types of datasets. Our results in section 4.3.1 also show that there are indeed significant differences not explainable by the pruning routine alone. We have also carried out experiments with the random split and the *PE* splitting rules and found much worse results, particularly bad for the random split (very long and badly performing trees even when pruned). On the other hand, it has been abundantly shown by comparative works on tree pruning (see namely [9]) that there are no safe pruning methods; all may lead to underpruning or overpruning depending on the dataset.

Tree pruning wouldn't in principle be needed if one employed a splitting rule favoring the most discriminative (higher significant *p*) features at an early stage and would stop tree growing as soon as a node with large class overlaps is reached. Now, the MEE algorithm given the influence of *PE* on *EE* shown in section 3, favors high discriminative features. At the same time, as we have also seen in section 2, the *EE* curves will lead to MEE split points at feature interval ends when the classes are overlapped; this often affords an easy method to stop tree growing before overfitting could creep in.

We now point out three reasons why the MEE algorithm is not very sensitive to pruning.

1. Test set error curves

Overfitting of tree classifiers can often be detected during the design phase by setting aside an independent test set and looking to the test set error rate during tree growing; overfitting will be revealed by an inflexion of the error curve, which goes upward after reaching a minimum error point. We have performed a large number of experiments with the MEE algorithm designing the tree with 85% of randomly chosen cases and testing it in the remaining 15%, and plotted the mean and mean ± standard deviation of the training and the test set error estimates along the tree level for 20 repetitions of the tree design. The results obtained are mostly exemplified by those of Figure 12; no symptoms of overfitting are apparent. In only a few datasets (3 out of 24) we obtained sometimes curves with overfitting symptoms (always in the last one or two levels).

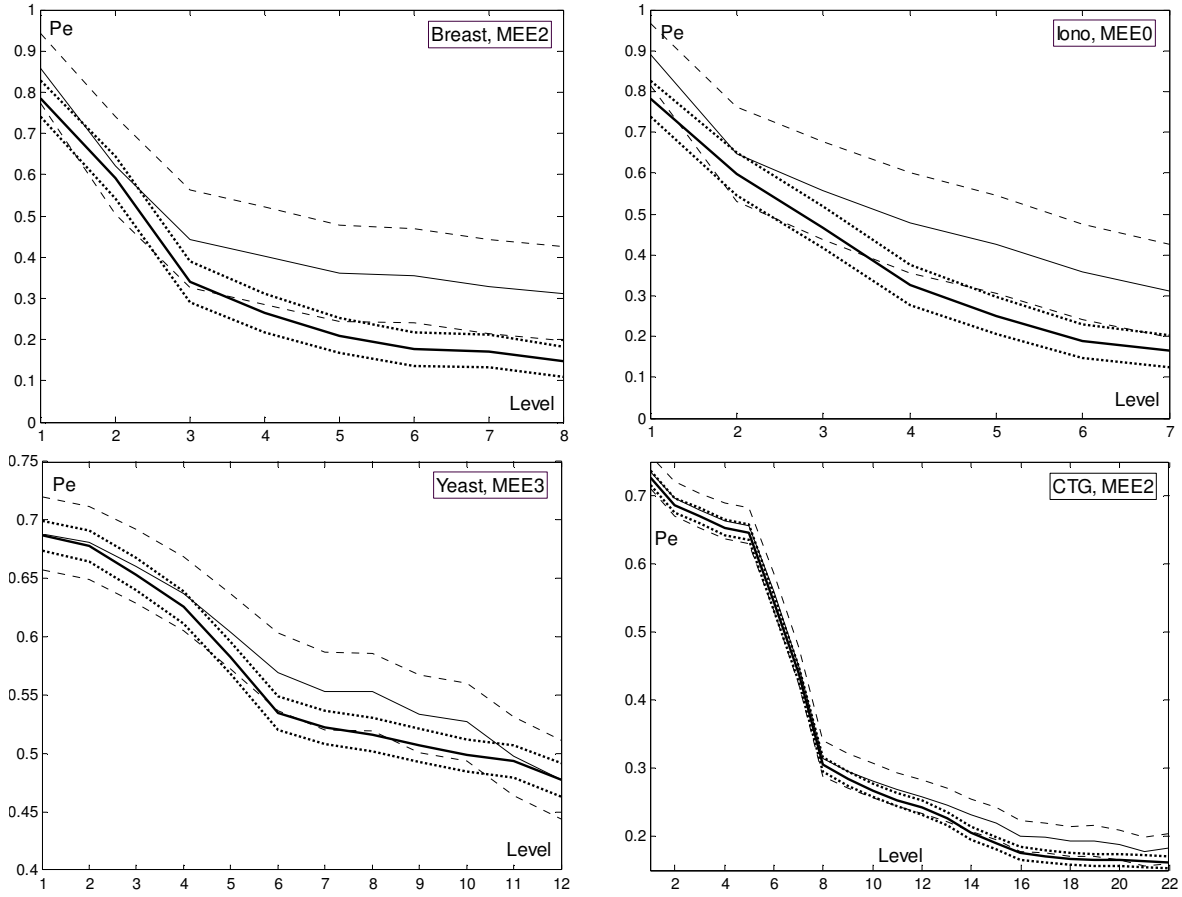

Fig. 12. Mean (solid line) and mean±std (dashed line) of the training set error (black, thick line) and test set error (grey, thin line) in 20 experiments on trees designed with 85% of the cases (randomly drawn) and tested in the remaining cases. The MEE suffix number means the number of merged classes in class unions (0, if no class unions are used).

2. Best level statistics

The pruning CCP routine used by the MEE algorithm allows one to know the best pruning level counted from the tree bottom. As shown in Table 4, 66.7% of the times the CCP routine didn't find the need to prune. Moreover, for the 'best level' equal to 1 or 2 we found that the tree solutions had equal or close error rates.

Table 4. Number of occurrences of best-level (pruning) values.

Best level	0	1	2	≥3
counts	16	3	2	3
%	66.7	12.5	8.3	12.5

3. Error rates with and without pruning

Table 5 presents the LOO estimates of the error rate obtained with the MEE algorithm with and without pruning for the 24 real-world datasets. A Wilcoxon signed rank test applied to the "w/" and "wo/" values confirms what simple visual inspection would have hinted: no significant difference ($p = 0.93$) between both groups of values.

Table 5. LOO estimates of MEE P_e with (w/) and without (wo/) pruning.

	Balance	Breast	Breast4	Car	Clev. HD2	Colon	CTG	Dermatol.
w/	0.3056	0.4340	0.1321	0.0764	0.2222	0.1613	0.1839	0.0726
wo/	0.2832	0.3962	0.1321	0.1146	0.2896	0.0645	0.1811	0.0615
	E-coli	E-coli4	Glass	Image Seg.	Ionosphere	Led	Leukemia	Page blks
w/	0.1407	0.0856	0.3084	0.0584	0.1311	0.3100	0.2222	0.0360
wo/	0.1407	0.0826	0.3037	0.0576	0.1311	0.3250	0.2222	0.0402
	Pb12	P. Diabetes	P. Gene	Spect-Heart	Thyroid	Wdbc	Yeast	Zoo
w/	0.1447	0.2760	0.1698	0.1436	0.0791	0.0668	0.5382	0.0990
wo/	0.1447	0.2760	0.2547	0.1685	0.0791	0.0668	0.5348	0.0990

6 Conclusions

The basic rationale of the MEE approach is that it searches for splits concentrating the error distribution at zero. For the classic impurity criteria (and other criteria as well) what the split is doing in terms of the node error distribution is unclear.

We have seen that the entropy-of-error function has a concave behavior resembling in this respect the classical impurity functions, Gini diversity index and information measure. There is, however, an important distinction as explained in detail in section 3.2: whereas these impurity functions are insensitive to the error rate the *EE* function is influenced by it and is thus able to make more reasonable decisions at node level, specifically with more balanced errors of competing classes. Moreover, we have also seen that the MEE criterion will not work for distributions that are too largely overlapped, providing a natural way of when to stop tree growing.

The experiments carried out in 24 real-world datasets show that the MEE approach to tree design competes well with the popular CART-Gini, CART-Information-Gain, CART-Twoing and C4.5 algorithms, producing in some cases improved solutions. The real strength of the MEE approach seems however to lie in the following four issues: short trees; better generalization; stable designs; reduced sensitivity to pruning.

In what concerns tree size we have provided experimental evidence that the MEE algorithm will often produce shorter trees than the other competing approaches (section 4.3.2). The difference was found to be statistically significant in post-hoc comparison tests when comparing the MEE against the C4.5 algorithm. (One must bear in mind that C4.5 is more intricate than the simple CART-style approach followed by MEE, producing highly performing trees at the expense of increased complexity.)

Regarding the generalization issue MEE trees seem to generalize better than (at least) those produced by C4.5 and CART-Twoing, when judging by the LOO and resubstitution error rate estimates obtained for the 24 datasets (section 4.3.1). This characteristic is of course a consequence of the shorter trees produced by the MEE algorithm.

During the process of obtaining LOO estimates of the error rate one is performing a (usually large) set of designs with largely overlapped training sets (in fact as large as possible). Stable design methods produce by definition very close solutions for mild input changes, implying necessarily small tree size variations in LOO experiments. Now, judging from the tree size ranges the LOO experimental evidence presented in section 4.3.2 ranks together the MEE and C4.5 algorithms as equally stable and significantly more stable than the other methods.

Finally, we have also presented results in section 5 showing that the MEE algorithm is not very sensitive to pruning, at least when cost-complexity pruning is used. With this pruning method we found that the solutions with or without pruning were not significantly different and as a matter of fact they were coincident in many cases.

References

1. Alon U et al. (1999) Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. PNAS, 96:6745-6750.
2. Argentiero P, Chin R, Beaudet P (1982) An Automated Approach to the Design of Decision Tree Classifiers. IEEE Tr. PAMI, vol. 4, 1:51-57.
3. Asuncion, A. & Newman, D.J. (2007). UCI Machine Learning Repository [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information and Computer Science.
4. Breiman L, Friedman JH, Olshen RA, Stone CJ (1993) Classification and Regression Trees. Chapman & Hall/CRC.
5. Buntine WL (1990) A Theory of Learning Classification Rules. PhD Thesis. Univ. of Technology, Sidney.
6. Buntine W, Niblett T (1992) A Further Comparison of Splitting Rules for Decision-Tree Induction. Machine Learning, 8, pp. 75-85.
7. Demšar J (2006) Statistical Comparisons of Classifiers over Multiple Data Sets. J. of Machine Learning Research, 7:1-30.
8. Devroye L, Giörfi L, Lugosi G (1996) A Probabilistic Theory of Pattern Recognition. Springer-Verlag.
9. Esposito F, Malerba D, Semeraro G (1997) A Comparative Analysis of Methods for Pruning Decision Trees. IEEE Tr. PAMI, vol. 19, 5, pp: 478-491.
10. Fu KS (1968) Sequential Methods in Pattern Recognition and Machine Learning. Academic Press.
11. Gelfand SB, Ravishankar CS, Delp EJ (1991) An Iterative Growing and Pruning Algorithm for Classification Tree Design. IEEE Tr. PAMI, vol.13, 2:163-174.
12. Geman D, Jedynak B (2001) Model-Based Classification Trees. IEEE Tr. IT, vol.47, 3:1075-1052.
13. Golub TR et al. (1999) Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science, 286:531-537.
14. Jacobs R, Jordan M, Nowlan S, Hinton G (1991). Adaptive Mixtures of Local Experts. Neural Computation, 3:79-87.
15. Kulkarni AV (1978) On the Mean Accuracy of Hierarchical Classifiers. IEEE Tr. C, vol. 27, 8:771-776.
16. Li R-H, Belford GG (2002) Instability of Decision Tree Classification Algorithms. Proc. 8th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining, pp. 570-575.
17. Loh W-Y, Shih Y-S (1997) Split Selection Methods for Classification Trees. Statistica Sinica, 7:815-840.
18. Marques de Sá JP (2007) Applied Statistics Using SPSS, STATISTICA, MATLAB and R (2nd edition). Springer-Verlag.
19. Marques de Sá JP, Gama J, Sebastião R, Alexandre LA (2009) Decision Trees Using the Minimum Entropy-of-Error Principle. Proc. CAIP 2009 (Xiaoyi Jiang, Nicolai Petkov eds), Springer LNCS, pp. 799-807.
20. Maszczyk T, Duch W (2008) Comparison of Shannon, Renyi and Tsallis Entropy used in Decision Trees. In L. Rutkowski et al. (Eds) Artificial Intelligence and Soft Computing-ICAISC 2008, pp. 643-651, Springer Verlag.
21. Mingers J (1989) An Empirical Comparison of Selection Measures for Decision-Tree Induction. Machine Learning, 3, pp. 319-342.
22. Quinlan JR (1986) Induction of Decision Trees. Machine Learning, 1: 81-106.
23. Quinlan JR (1990) Decision Trees and Decisionmaking. IEEE Tr. Syst. Man and Cybernetics, vol.20, 2:339-346.
24. Quinlan JR (1993) C4.5 Programs for Machine Learning. Morgan Kaufmann
25. Raileanu LE, Stoffel K (2004) Theoretical Comparison Between the Gini Index and Information Gain Criteria. Annals of Mathematics and Artificial Intelligence, 41: 77-93.
26. Safavian SR, Landgrebe D (1991) A Survey of Decision Tree Classifier Methodology. IEEE Tr. SMC, vol. 21, 3: 660-674.
27. Silva L, Felgueiras CS, Alexandre L, Marques de Sá J (2006) Error Entropy in Classification Problems: A Univariate Data Analysis. Neural Computation, 18, 2036-2006.
28. Silva LM, Embrechts M, Santos JM, Marques de Sá J (2008) The Influence of the Risk Functional in Data Classification with MLPs. Proc. ICANN 2008 (eds. V. Kurková et al.), Springer LNCS, pp. 185-194.
29. Stoller D (1954) Univariate Two-Population Distribution-Free Discrimination. Journal of the American Statistical Association, 49:770-777.
30. Swain PH (1977) The Decision Tree Classifier: Design and Potential. IEEE Tr. GE, vol.15, 3:142-147.