darkred!85!blue!60!black

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe de Albuquerque Mello Pereira

Binary Splitting Criteria for Large Categorical Attributes in Decision Trees

Dissertação de Mestrado

Thesis presented to the Informática of the Departamento de Informáticado Centro Técnico Científicoda PUC-Rio as partial fulfillment of the requirements for the degree of Doutor

Advisor: Prof. Eduardo Sany Laber

Rio de Janeiro
February 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe de Albuquerque Mello Pereira

Binary Splitting Criteria for Large Categorical Attributes in Decision Trees

Thesis presented to the Postgraduate Program in Informática of the Departamento de Informática, PUC-Rio as partial fulfillment of the requirements for the degree of Mestre em Informática.

Prof. Eduardo Sany Laber
Advisor
Departamento de Informática — PUC-Rio

Prof. Hélio Côrtes Vieira Lopes
Departamento de Informática — PUC-Rio

Prof. Marco Serpa Molinaro
Departamento de Informática — PUC-Rio

Prof. Márcio da Silveira Carvalho
Vice Dean of Graduate Studies
Centro Técnico Científico da PUC-Rio

Rio de Janeiro, February 28th, 2018

Felipe de Albuquerque Mello Pereira

Bachelor's in Electrical Engineering and Pure Mathematics at the Pontifícia Universidade Católica do Rio de Janeiro (2010 and 2011). Masters' in Mathematics at the Pontifícia Universidade Católica do Rio de Janeiro (2013).

Bibliographic data

Pereira, Felipe de Albuquerque Mello

Binary Splitting Criteria for Large Categorical Attributes in Decision Trees / Felipe de Albuquerque Mello Pereira ; advisor: Eduardo Sany Laber. — 2018.
36 f. : il. ; 30 cm

Dissertação (Mestrado em Informática)-Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2018.
Inclui bibliografia

1. Informática – Teses. 2. Árvores de Decisão; Problema de Corte Máximo; Algoritmos Aproximativos. I. Laber, Eduardo Sany. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

TODO: acknowledgment.

Thanks to CNPq for the conceded scholarship during my Masters.

Abstract

Pereira, Felipe de Albuquerque Mello; Laber, Eduardo Sany (advisor). **Binary Splitting Criteria for Large Categorical Attributes in Decision Trees**. Rio de Janeiro, 2018. 36p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

TODO: abstract.

Keywords

Decision Trees; Max-cut Problem; Approximated Algorithms

Resumo

Pereira, Felipe de Albuquerque Mello; Laber, Eduardo Sany. **Critérios de Splits Binários para Atributos Categóricos Grandes em Árvores de Decisão.**. Rio de Janeiro, 2018. 36p. Dissertação de Mestrado — Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

TODO: resumobr.

Palavras-chave

Árvores de Decisão; Problema de Corte Máximo; Algoritmos Aproximativos

Contents

List of Figures

List of Tables

1
Introduction

Decision Trees and Random Forests are among the most popular methods for classification tasks. Decision Trees, specially small ones, are easy to interpret, while Random Forests usually yield more accurate classifications. One of the key issues in these methods is how to select an attribute to associate with a node of the tree/forest. An important related issue is how to split the samples once the attribute is selected.

There is a number of papers discussing aspects related with attribute selection, such as: how to design criteria to evaluate the quality of different types of attributes; whether binary or multi-way splits shall be used and how to remove bias from splitting criteria. For recent surveys on this topic we refer to **???**.

Many criteria, with different properties, have been proposed to evaluate the quality of different types of attributes, including continuous and categorical ones. Among the most popular criteria, we have the Gini Gain and the Information Gain .

Despite the large body of work we believe there are still questions to be answered. One of them is to how to properly handle nominal attributes that may assume a large number of values. Before explaining the reason behind our statement we would like to remark that this kind of attribute appears naturally in some applications (e.g.: states of a country or letters from some alphabet). In addition, they may arise as the result of aggregating attributes that have few distinct values with the goal of capturing possible correlation between them, as pointed out by **?**. As an example, consider 5 binary attributes (e.g. medical tests) and a target binary variable that has large probability of being positive if at least 3 out of the 5 binary tests are positives. By aggregating the 5 binary variables we obtain a new attribute with $2^5 = 32$ values that captures this relation. If we used the 5 attributes separately we would need 5 levels in the tree to be able to capture the relation between them and the target class, thus incurring a large fragmentation of the set of samples.

To properly face multi-valued nominal attributes we have to deal with the computational time required to compute good splits. Our contribution, explained in the next section, is related with this issue.

A brute force search to compute the best binary split requires $\Omega(2^n)$ time, where n is the number of distinct values the attribute may assume. The computational efficiency can be improved if a n-ary split is used rather than a binary one. However, this may lead to a severe fragmentation of the sample space, which is not desirable: the number of samples available for each of the children of the split node may be small and, as a consequence, the underlying classification tasks may become significantly more difficult. When the target variable is binary, the Gini Gain, proposed in the influential monography by Breiman et al **?**, can be computed efficiently. However, when the number of classes k is larger than 2, most, if not all, of the available exact solutions take exponential time in (n, k). The Twoing method **?**, which is equivalent to Gini Gain when $k = 2$, is an interesting case since its running time is $O(2^{\min\{n,k\}})$ rather than $O(2^n)$.

When both n and k are large, in the sense that an exhaustive search does not run in a reasonable time, one can rely on heuristics to compute the best binary split. As an example, the GUIDE algorithm **?**, the last of a series of algorithms/developments designed by Loh and its contributors, deals with a nominal variable X as follows: if $k = 2$ or $n \leq 11$ the Gini Index is computed; if $k \leq 11$ and $n > 20$ a new variable X' with at most k distinct values is created according to a certain rule and an exhaustive search is performed over it; finally, if $k > 11$ or $n \leq 20$, X is binarized and a Linear Discriminant Analysis (LDA) is employed. These rules reflect the difficulty in dealing with multi-valued nominal attributes. In general, the main drawback of using heuristics is the lack of a theoretical guarantee about their behavior.

1.1 Our Contribution

Given this scenario, in chapter 3 we propose a framework for designing criteria, with nice theoretical properties, for evaluating the quality of multi-valued nominal attributes. Criteria generated according to this framework run in polynomial time in both the number of values and classes and have a theoretical guarantee that they are close to optimal. The key idea consists of formulating the problem of finding the best binary partition for a given attribute A as the problem of finding a cut with maximum weight in a complete graph whose nodes are associated with the values that A may assume and the edges' weights capture the benefit of putting values in different partitions. The motivation behind the use of the max-cut problem is the existence of efficient algorithms with approximation guarantee, in particular the one proposed by **?**, with 0.878 approximation, and local search algorithms with 0.5-approximation

?.

We discuss two criteria that are derived from this framework: the first one can be seen as a natural variation of the Gini Gain, while the second criterion uses the χ^2-test to set the edges' weights. For that, each edge e_{ij}, between nodes v_i and v_j, is thought as a binary attribute $A(i, j)$, with values v_i and v_j. After discussing these criteria, we show how to extend them to handle numeric attributes.

We also present a number of experiments that suggest that one of our criteria is competitive with the Twoing method, which is – as far as we know – the only well-established criterion with binary splits that can be optimally computed for large n when $k > 2$. However, in contrast with our methods, Twoing cannot handle datasets that also have a large number of classes. In addition, the experiments also provide evidence of the potential of aggregating attributes for improving the accuracy of decision trees.

1.2 Organization

In chapter 2 we explain how decision trees are used for classification problems and how they are constructed. We also present the main impurity measures and splitting criteria used in the literature, together with their execution-time complexity.

Chapter 3 contains the framework for generating splitting criteria that run in polynomial time. Its relation with the Max-Cut problem and its approximation algorithms are explained and some criteria obtained from this framework are presented.

Later, in chapters **??** and **??**, we compare all the existing criteria and see how they perform in practice. In chapter **??** we explore how the many heuristics used to find splits with optimal impurity perform compare with the exact methods. This suggests a couple of criteria that perform better and can be used when the number of values and classes are high. In chapter **??** we analyse these heuristics together with exact methods on real datasets that contain attributes with large number of values and classes. Lastly, in chapter **??** we present our study conclusions.

2
Background

2.1
Notation

We adopt the following notation throughout the dissertation. Let S be a set of N samples and $C = \{c_1, \ldots, c_k\}$ be the domain of the class label. In addition, for an attribute A, we use $A(s)$ to denote the value taken by attribute A on sample s; we use $V = \{v_1, \ldots, v_n\}$ to denote the set of values taken by A; A_{ij} to refer to the number of samples from class c_j for which A takes value v_i; N_i for the number of samples with value v_i for attribute A and S_j for the number of samples from class c_j. Furthermore, we let $p_j = S_j/N$ and $p_{ij} = Pr[C = c_j | A = v_i]$. We observe that the estimator of maximum likelihood for p_{ij} is A_{ij}/N_i.

2.2
Impurity Measures

Many of the splitting criteria follow the same algorithm:

TODO: algorithm create$_tree(S : setofsamples, L : listofattributesinformation, I :$ $impuritymeasure)ifSdoesnotmeetthestoppingcriterion : Foreachattribute :$ $getthevalues\acute{s}plitthatyieldsthesmallestimpurity(measuredbyI)SplitSusingtheattributewhosebestv$

Therefore a good place to start is by presenting the two most common impurity measures found in the literature.

TODO: falar de contingency tables aqui.

2.2.1
Gini

The Gini Index for a set of samples S is given by

$$Gini(S) = 1 - \sum_{i=1}^{k} (p_i)^2. \tag{2-1}$$

The Gini Index can be used to generate binary splits and, as a consequence, binary decision trees.

The Gini Gain, Δ_G, induced by a binary partition (L, R) of the set of values V is given by

$$\Delta_G(L, R) = Gini(S) - p_L Gini(S_L) - p_R Gini(S_R), \tag{2-2}$$

where $S_L = \{s \in S | A(s) \in L\}$, $S_R = \{s \in S | A(s) \in R\}$, $p_L = |S_L|/N$ and $p_R = |S_R|/N$. Therefore, the largest the Gini Gain is, the better the partition.

2.2.2 Entropy

The Entropy for a set of samples S is given by

$$Entropy(S) = -\sum_{i=1}^{k} p_i \log(p_i) \tag{2-3}$$

The Entropy can be used to generate binary splits and, as a consequence, binary decision trees.

The Information Gain IG induced by a binary partition (L, R) of the set of values V is given by

$$IG(L, R) = Entropy(S) - p_L Entropy(S_L) - p_R Entropy(S_R), \tag{2-4}$$

where $S_L = \{s \in S | A(s) \in L\}$, $S_R = \{s \in S | A(s) \in R\}$, $p_L = |S_L|/N$ and $p_R = |S_R|/N$. Therefore, the largest the Information Gain is, the better the partition.

2.3 Splitting Criteria

In this section we recall some well-known splitting criteria.

2.3.1 Gini Gain

This criterion generates all 2^n binary valueśplit and the partition with maximum Gini Gain shall be selected. As shown in **?**, for the 2-class problem this optimal partition can be computed in $O(n \log n + N)$ time: sort the values by the frequency of class 0 on them. The best binary split of S will be given by one of the values splits that follow this order (TODO: explicar melhor).

For problems with more than 2 classes, however, there is no efficient procedure with theoretical approximation guarantee to compute the Gini Gain in subexponential time in n.

2.3.2 Twoing

The Twoing criterion for a binary partition (L, R) of the set of values V is given by

$$0.25 \cdot p_L \cdot p_R \cdot \left(\sum_{i=1}^{k} |p_L^i - p_R^i| \right)^2,$$

where
$p_L^i = \frac{|\{s \in S_L : s \text{ belongs to class } c_i\}|}{|S_L|}$ and $p_R^i = \frac{|\{s \in S_R : s \text{ belongs to class } c_i\}|}{|S_R|}$.

When the Twoing criterion is used to generate binary decision trees, the binary partition with maximum twoing shall be selected at each node.

As shown in **?**, such partition can be calculated in $O(N + \min\{n \log n 2^k, 2^n\})$ time by considering all possibilities of partitioning the classes into two superclasses and applying the Gini Gain criterion on each of them. We shall remark that, for the 2-class problem, the Twoing criterion and the Gini Gain compute the same binary partitions.

2.3.3 Information Gain

This criterion works exactly the same as the Gini Gain, but replacing the Gini impurity by the Entropy. First it generates all 2^n binary valuesśplit and the partition with maximum Information Gain shall be selected. For the 2-class problem, the same result valid for the Gini Gain works here, and the optimal partition can be computed in $O(n \log n + N)$ time. Once again, when the number of classes is larger than 2 there is no efficient procedure with theoretical approximation guarantee to compute the Information Gain in subexponential time in n.

A related criterion is the Gain Ratio, where the Information Gain of an attribute is normalized by the potential information of that attribute. This is used as a way of decreasing the bias of the k-ary Information Gain criterion towards attributes with larger number of values. Since we are only interested in binary splits in this dissertation, we will not go into its details.

2.3.4 χ^2 criterion

The χ^2 is a popular criterion that was used in **?**. It is also the first one shown here not based on impurity measures, and it only works for k-ary (instead of binary) splits. Is is mentioned here because of its relation to the framework presented in chapter 3.

For an attribute A the χ^2 criterion is given by

$$\sum_{i=1}^{n}\sum_{j=1}^{k}\frac{(A_{ij}-E[A_{ij}])^2}{E[A_{ij}]}, \tag{2-5}$$

where $E[A_{ij}] = N_i p_j$.

2.3.5
Conditional Inference Trees

Conditional Inference Trees are actually a framework of creating criteria that are bias-free when it comes to the number of values in an attribute. It was published by **?** and still is the only known method of obtaining criteria that do not have any bias towards attributes with larger number of values.

It first chooses the best attribute to split at the current node and then evaluates all possible binary splits using any given impurity measure, choosing the best one found.

TODO: ver se notacao bate.

To choose the attribute in which to split, first one has to calculate the conditional expectation μ_j and covariance Σ_j of the permutation test of all attributes A_j. Then, in order to compare the attributes, we need to calculate the p-value of a univariate test statistics c_{quad} calculated on μ_j and Σ_j. The only exact form of doing this is by using the quadratic form (TODO: add equation and ref), which it follows an asymptotic χ^2 distribution with degrees of freedom given by the rank of Σ_j. Since this involves the calculation of a pseudo-inverse, this is usually time consuming (the time complexity of the pseudo-inverse is cubic on the dimension of Σ_j, which is $n_j * k$).

TODO: colocar formulas

This method, although very complicated and somewhat slow, is used by the community very much. We mentioned it here because it will be used in our experiments in chapter **??**, since measuring the performance of any impurity measure in it is important in order to evaluate its usefulness. (TODO: melhorar texto desse paragrafo)

2.4
Heuristics for Splitting Decision Tree Nodes

As seen in the previous section, calculating the optimal split takes exponential time in the number of values or classes. Therefore many heuristics were created to create decision trees in this situation. The most used ones are listed below. All of them work with any impurity measure (e.g.: Gini or Entropy), but some of them work best with one of them. When this is the case, it will be mentioned.

2.4.1 SLIQ and SLIQ-ext

SLIQ was presented in (TODO: add citation) and it's a very simple greedy heuristic. Given an attribute, one starts with all the values going to the left split, and none on the right split. We then choose a value to go from the left to the right split. This value is the one that, when changing from the left to the right sides, decreases the impurity (increases the impurity gain) the most. This is repeated until there is no way of moving a value from the left to the right and decreasing the impurity.

SLIQ-ext is a simple extension, where we keep changing values from the left to the right until the left side is empty (that is, we move from the left to the right even if that increases the impurity). Once again the value to move is chosen in a greedy fashion. SLIQ-ext returns the values' split seen that had the lowest impurity.

2.4.2 PC and PC-ext

These heuristics are based on the Principal Component of the contingency table the and were presented in (TODO: add citation). One first calculates the class probability distribution of every value, which is done by normalizing the contingency table rows to have 1 in the sum norm. We then group values into "supervalues" where each value in the same supervalue has the same class probability distribution. Now we get the contingency table of these supervalues and calculate the first principal component of this matrix. One then calculates the inner product of each class probability vector of the supervalues with the principal component and sort the supervalues by it. We then calculate the supervalues split's impurity gain given by splits of the form

TODO: write formula left, right.

Once we find the supervalues split with the largest impurity gain, we translate the supervalues into original values to obtain a valid partition.

PC-ext is a simple extension of this algorithm, where instead of only testing the supervalues splits given by (TODO: ex ref), we also test the split given by it and exchanging the last supervalue on the left with the first supervalue on the right (where first and last are given by the order after calculating the inner product).

2.4.3
Largest Alone

First one calculates the most frequent class and group the other classes in a single superclass. We then apply the Gini Gain criterion on this two-class problem. Since calculating the class frequencies can be done using the contingency table, this heuristic takes $O(N + n * k + n * \log(n))$ time in total.

Another advantage of the Largest Alone heuristic is that is has an approximation guarantee of 2 (TODO: conferir) to the best gini impurity of the original problem. It is also proved that there is no other way of grouping classes into superclasses that has a smaller approximation guarantee for the Gini impurity. It can also be used with the Information Gain, instead of Gini Gain, but its approximation guarantee increases to 3 (TODO: conferir). These bounds are proved in (TODO: ver como colocar referencia).

2.4.4
List Scheduling

First one calculates the frequency of every class. Then, we use a List Scheduling algorithm to group the classes into 2 superclasses as balanced as possible (in terms of number of samples). Lastly we apply the Information Gain criterion on this two-class problem. Again, since calculating the class frequencies can be done using the contingency table, this heuristic and the List Scheduling algorithm is linear in the number of classes, this heuristic also takes $O(N + n * k + n * \log(n))$ time in total.

Similarly to the Largest Alone heuristic, the List Scheduling heuristic has an approximation guarantee of 2 (TODO: conferir) to the best entropy impurity of the original problem. This is proved in (TODO: ver como colocar referencia). It is also proved that, for the entropy impurity, the best form of grouping classes into superclasses is by balancing them the best way possible.

3
Framework for Generating Splitting Criteria for Multi-valued Attributes

First we recall some definitions and results for the Max-Cut problem. These definitions will be used in the following section, when we define our framework.

3.1
The Maximum Weighted Cut Problem

We recall some definitions from graph theory. A cut X in a weighted graph $G = (V, E)$ is a subset of vertexes of V. The weight of a cut X, denoted here by $w(X)$, is the sum of the weights of the edges that have one endpoint in X and the other one in $V - X$.

The problem of computing the cut X^* with maximum weight in a graph with non-negative weights is NP-Hard. However, there are good approximation algorithms available. A remarkable one is the randomized algorithm proposed in **?** that relies on a formulation of the max-cut problem via semidefinite programming. This algorithm, denoted here by GW, returns a cut X that satisfies $E[w(X)] \geq 0.878w(X^*)$.

TODO: explicar algoritmo

Another possibility to solve the Max-Cut problem is by using the `GreedyCut` algorithm, presented in Algorithm 1. It obtains a cut X such that $w(X) \geq 0.5w(X^*)$ **?**. The algorithm starts with two empty sets X and X'. Then, it scans the nodes and assigns each of them to the set that provides the maximum improvement on the weight of the current cut (ties are broken arbitrarily).

The solutions obtained by both GW and `GreedyCut` can be improved via a local search. In its simplest version, it moves a node from one group to the other while some improvement on the cut weight is possible. Although this algorithm is not polynomial in the worst case, it has polynomial behavior in the smoothed analysis framework **?**. In addition, it is always possible to set a limit on the number of moves. A more refined version allows exchanging a pair of nodes as long as the weight of the cut is improved. In our experiments we use the version presented at Algorithm 2.

Algorithm 1 GreedyCut(V: set of nodes)

$X \leftarrow \emptyset; X' \leftarrow \emptyset$
for $j = 1, .., n$ **do**
If

$$\sum_{v \in X} w(v_i, v) > \sum_{v \in X' - V} w(v_i, v)$$

add v_i to X' **Else** add v_i to X
end for
Return X and X'

Algorithm 2 LocalSearch(X, X'): set of nodes

label : *loop_start*
for $i = 1, ..., n$ **do**
if switching v_i's side improves cut weight **then**
switch v_i and update cut weight, X, X'
goto loop_start
end if
end for
for pair $(v_i, v_j) \in X \times X'$ **do**
if switching v_i and v_j improves cut weight **then**
switch v_i and v_j, update cut weight, X, X'
goto loop_start
end if
end for
Return X, X'

3.2 A Framework for Generating Splitting Criteria

In this section we explain our approach to building binary splitting criteria for multi-valued nominal attributes.

Let A be a nominal attribute that takes values in the domain $V = \{v_1, \ldots, v_n\}$. Our framework to produce a splitting criterion I consists of three steps:

1. Create a complete graph $G = (V, E)$ with n vertexes.

2. Assign a non-negative weight w_{ij} to the edge that connects v_i to v_j. This value shall reflect the benefit of putting v_i and v_j in different partitions. Different definitions of w_{ij} yield to different criteria, as we explain further.

3. Ideally, the value of the criterion I for attribute A is the weight of the cut with maximum weight in G. However, this is not a reasonable possibility for large n since the problem of computing the cut X^* with maximum weight in a graph with non-negative weights is NP-Hard. Thus, the

value of criterion I is given by the weight of the cut obtained by some algorithm, with approximation guarantee, for the maximum cut problem in G.

What distinguishes the criteria generated by our framework is how the weights of the edges are set and the method employed to compute the cut on graph G. Here, we discuss two ways to set the weights: the first one yields to criteria that are related with the Gini Gain, while the second is built upon some given splitting criterion that works well for binary attributes.

3.2.1 The Squared Gini Criterion

Here, we discuss how to set the weights so that we obtain a criterion that can be seen as a variation of the Gini Gain discussed in Section **??**.

In fact, Lemma 1 below show that it is possible to define the weights of the edges so that

$$w(S_L) = Gini(S) - p_L^2 \cdot Gini(S_L) - p_R^2 \cdot Gini(S_R) \qquad (3\text{-}1)$$

for every partition (L, R) of V.

Note that the weight of the cut S_L in the above identity is similar to the expression for the Gini Gain given by equation (2-2). The difference is that p_L and p_R are replaced with p_L^2 and p_R^2, respectively. Because of the squares, this new criterion tends to favor more balanced partitions.

For that the proof of Lemma 1, recall that A_{ix} is the number of samples of class x that have value v_i, and C is used to denote the set of classes.

Lemma 1 *For every i, j, with $i \neq j$ and $i, j \in \{1, \ldots, n\}$, let*

$$w_{ij} = \frac{2 \sum_{\substack{x,y \in C \\ x \neq y}} A_{ix} A_{jy}}{N^2}.$$

Then, for every partition (L, R) of V we have $w(S_L) = Gini(S) - p_L^2 \cdot Gini(S_L) - p_R^2 \cdot Gini(S_R)$.

Proof: Let $S_{x,L}$ and $S_{y,R}$ be the number of samples of classes x and y in groups L and R, respectively. Moreover, let N_L and N_R be the number of samples in L and R, respectively. It follows from equation (2-1) that

$$N^2 Gini(S) = N^2 - \sum_{x=1}^{k} (S_{x,L} + S_{x,R})^2$$

$$N_L^2 Gini(S_L) = N_L^2 - \sum_{x=1}^{k} S_{x,L}^2$$

and

$$N_R^2 Gini(S_R) = N_R^2 - \sum_{x=1}^{k} S_{x,L}^2.$$

Since $N = (N_L + N_R)$ it follows that

$$N^2 Gini(S) - N_L^2 Gini(S_L) - N_R^2 Gini(S_R) =$$

$$2N_L N_R - 2\sum_{x \in C} S_{x,L} S_{x,R} =$$

$$2\sum_{x \in C} S_{x,L} \sum_{x \in C} S_{x,R} - 2\sum_{x \in C} S_{x,L} S_{x,R} =$$

$$2\sum_{\substack{x \neq y \\ x,y \in C}} S_{x,L} S_{y,R} = 2\sum_{\substack{x \neq y \\ x,y \in C}} \left(\sum_{i \in L} \sum_{j \in R} A_{ix} A_{jy} \right) =$$

$$N^2 \sum_{i \in L} \sum_{j \in R} w_{ij} = N^2 w(S_L)$$

Dividing the first term and the last term by N^2 in the above expression and, using $N_L = p_L \cdot N$ and $N_R = p_R \cdot N$, we establish the lemma. ■

It is worth mentioning that symmetric misclassification costs can be easily introduced in this case. In fact, let $mix(x, y)$ be the cost of mixing samples from classes x and y. We can define

$$w_{ij} = \sum_{\substack{x,y \in C \\ x \neq y}} mix(x, y) p_{ix} p_{jy}.$$

This measure favors the separation of the classes that incur a large cost in the case they are mixed.

3.2.2 Setting weights according to binary criteria

Our second way of defining the weights makes use of some given splitting criterion for binary nominal attributes. Such criterion is used to measure the quality of separating samples with value v_i from those with value v_j, for each i and j, and, thus, defining the edges' weights. Here, we investigate the criterion obtained by defining w_{ij} as the value of the χ^2-test for the attribute A when evaluated over the restricted dataset that contains only the samples of S with values v_i and v_j. In formulae,

$$w_{ij} = \sum_{\ell=1}^{k} \frac{(A_{i\ell} - E[A_{i\ell}])^2}{E[A_{i\ell}]} + \sum_{\ell=1}^{k} \frac{(A_{j\ell} - E[A_{j\ell}])^2}{E[A_{j\ell}]}$$

where $E[A_{i\ell}] = N_i p_\ell$ and $E[A_{j\ell}] = N_j p_\ell$.

In order to reduce the bias towards attributes with many values, we divide w_{ij} by $n-1$, for every pair i, j. We make this adjustment because each value contributes to the weight of $n-1$ edges.

We shall remark that, although not explored in this work, other criteria such as Information Gain or Gini Index could be used, instead of χ^2 test, to set the weights.

3.2.3
Handling Numeric Attributes

We observe that criteria from our framework can handle a numeric attribute A with t distinct values $v_1, \ldots, v_t$ by considering it as collection of $t-1$ binary attributes, where the j-th attribute, A^j, splits the samples into the groups $\{s|A(s) \leq v_j\}$ and $\{s|A(s) > v_j\}$. The split obtained by criterion I on a numeric attribute A matches the split of the best attribute among $A^1, \ldots, A^{t-1}$, according to I.

4
Experiments on Splits

In this chapter we compare the ability of different heuristics in finding the values' split with lowest impurity.

5
Experiments on Real Datasets

In this chapter we describe our experimental study on real datasets. First, we describe the chosen datasets. Next, we discuss the max-cut algorithms employed and, then, we present our results.

All experiments described in the following sections were executed on a machine with the following settings: Intel(R) Core(TM) i7-4790 CPU @ 3.60GHz with 32 GB of RAM. The code was developed using Python 3.6.1 with the libraries numpy, scipy, scikit-learn and cvxpy. The project can be accessed in `github.com/felipeamp/max_cut_paper`. It includes the code, the datasets and the results of our experiments.

5.1
Datasets

We employed 11 datasets in total. Eight of them are from the UCI repository: Mushroom, KDD98, Adult, Nursery, Covertype, Cars, Contraceptive and Poker **?**. Two others are available in Kaggle: San Francisco Crime and Shelter Animal Outcome **?**Austin Animal Center (2016). The last dataset was created by translating texts from the Reuters database **?** into phonemes, using the CMU pronouncing dictionary **?**.

We chose these datasets because they have at least 1000 samples and they either contain multi-valued attributes or attributes that can be naturally aggregated to produce multi-valued attributes. From the KDD98 dataset we derived the datasets KDD98-k, for $k = 2$ and 9. These datasets contain only the positive samples (people that donate money) of KDD98 and the target attribute, Target_D, is split into k classes, where the i-th class correspond to the i-th quantile in terms of amount of money donated. For the Reuters Phonemes dataset, we extracted 10000 samples containing the 15 most common phonemes as class and try to predict when they are about to happen given the 3 preceding phonemes. This dataset is motivated by Spoken Language Recognition problems, where phonotactic models are used as an important part of the classification system **?**. For the San Francisco Crime dataset, we give the month, day of the week, police department district and latitute/longitude and try to predict the crime category. Lastly, for the Shelter Animal Outcomes dataset,

Table 5.1: Information about the employed datasets after data cleaning and attributes aggregation. Column k is the number of classes and `Reg` stands for Regression; columns m_{nom} and m_{nom}^{ext} are the number of nominal attributes in the original and the extended datasets (when it exists), respectively; column m_{num} is the number of numeric attributes.

Dataset	Samples	k	m_{nom}	m_{nom}^{ext}	m_{num}
`Mushroom`	5644	2	22	N/A	0
`Adult`	30162	2	8	N/A	6
`KDD98`	4843	`Reg`	65	N/A	314
`Nursery`	12960	5	8	11	0
`CoverType`	581012	7	44	46	10
`Car`	1728	4	6	8	0
`Contracep`	1473	3	7	9	2
`Poker`	25010	10	10	0	0
`Shelter`	26711	22	5	N/A	1
`S.F. Crime`	878049	39	3	N/A	2
`Phonemes`	10000	15	3	N/A	0

we converted the age into a numeric field containing the number of days old and separated the breed into two categorical fields, repeating the breed in both in case there was only one originally. We also removed the AnimalID, Name and the DateTime. For this dataset we try to predict the outcome type and subtype (concatenated into a single categorical field). For both San Francisco Crime and Shelter Animal Outcomes datasets we created a version of them (`S.F. Crime-15` and `Shelter-15`), containing only 15 classes, instead of the 39 and 22 original ones, respectively. This was done by grouping the rarest classes into a single one.

We also created extended versions of some of the above datasets by adding nominal attributes obtained by aggregating some of the original ones, as we detail below. Our goals are examining the impact of multi-valued attributes in the classification performance and also understanding how the different splitting criteria handle them.

Table 5.2 illustrates this construction.

– `Nursery-Ext`. This dataset is obtained by adding three new attributes to dataset Nursery. The first attribute has 15 distinct values and it is constructed through the aggregation of 2 attributes from group EMPLOY, one with 5 values and the other with 3 values. The second attribute has 72 distinct values corresponding to the aggregation of attributes from the attributes in group `STRUCT_FINAN`. The third attribute, with 9 distinct values, is the combination of the attributes in group `SOC_HEALTH`.

parents	has_nurs	Aggregated Attribute
usual	proper	usual-proper
usual	less_proper	usual-less_proper
usual	improper	usual-improper
usual	critical	usual-critical
usual	very_crit	usual-very_crit
pretentious	proper	pretentious-proper
pretentious	less_proper	pretentious-less_proper
pretentious	improper	pretentious-improper
pretentious	critical	pretentious-critical
pretentious	very_crit	pretentious-very_crit
great_pret	proper	great_pret-proper
great_pret	less_proper	great_pret-less_proper
great_pret	improper	great_pret-improper
great_pret	critical	great_pret-critical
great_pret	very_crit	great_pret-very_crit

Table 5.2: Aggregation of attributes parents and has_nurse from dataset `Nursery`

– `Covertype-Ext`. We combined 40 binary attributes related with the soil type into a new attribute with 40 distinct values. The same approach was employed to combine the 4 binary attributes related with the wilderness area into a new attribute with 4 distinct values. This is an interesting case because, apparently, the 40 (soil type) binary attributes as well as the 4 (wilderness area) binary attributes were derived from a binarization of two attributes, one with 40 distinct value and the other with 4 distinct values.

– `Cars-Ext`. To obtain this dataset, the 2 attributes related with the concept `PRICE`, `buying` and `maint`, were combined into an attribute with 16 distinct values. Moreover, the 3 attributes related with concept `CONFORT` were combined into an an attribute with 36 distinct values.

– `Contraceptive-Ext`. The 2 attributes related with the couple's education were combined into an attribute with 16 distinct values. Moreover, the 3 attributes related with the couple's occupations and standard of living were aggregated into a new attribute with 32 distinct values.

Samples with missing values were removed from the datasets. Table 5.1 provides some statistics.

5.2
Computing the Maximum Cut

The GW algorithm requires the solution of a semidefinite program (SDP), which may be computationally expensive despite its polynomial time worst case behavior. As an example, for an attribute with 100 distinct values, the solution of the corresponding SDP takes in average about 2 second in our machine. One the one hand, this is a tiny amount of time compared with that required to perform an exhaustive search on the 2^{100} possible binary partitions. On the other hand, faster alternatives are desirable, even at the cost of losing part of the theoretical approximation guarantee.

To avoid solving a SDP, we also evaluated a procedure that first executes the `GreedyCut` algorithm presented in Section 3.1 and then runs a local search as described in the Algorithm 2 in the same section. The use of this approach combined with the two ways of setting the edges' weights lead to Greedy LocalSearch SquaredGini (GLSG) and Greedy LocalSearch χ^2 (GLχ^2) criteria, respectively. For attributes with 100 distinct values this approach is 60-70 times faster than the one based on the GW algorithm.

In fact, experiments similar to those described in the next section show that GL approach consistently obtains better results than the GW algorithm in terms of both speed and accuracy. This advantage is likely related with the fact that the weights of the cuts computed by the GL approach are, in general, larger than those obtained by the GW algorithm. This pattern was observed in a set of experiments with the max-cut instances induced by the attributes of the datasets used in the experiments section. Due to these results, we proceed our investigation using only the GL approach.

It is worth mentioning that the greedy algorithm that precedes the local search guarantees that we still have a theoretical guarantee of being at least 0.5 of the optimal solution. The local search may be not

In addition, it was recently proved that a local search runs in polynomial time .... TODO: melhorar

5.3
Experimental Results

We performed a number of experiments to evaluate how the proposed methods behave with real datasets. All experiments consist of building decision trees with a predefined maximum depth. In addition, to prevent the selection of non-informative nominal attributes, we used a χ^2-test for each attribute at every node of the tree: if the χ^2-test on the contingency table of attribute A has p-value larger than 10% at a node ν, then A is not used in ν. Furthermore, attributes with less than 15 samples associated with its second most frequent value are also not considered for splitting. This helps avoid data overfitting.

Table 5.3.(a) presents the results of an experiment to compare the accuracy of Decision Trees built by our methods with those built by Twoing. In this experiment, the maximum depth was set to 16 and we considered just the nominal attributes of the datasets. Each accuracy is the average of 20 stratified 3-fold cross-validations, each generated with a different seed. The entry associated with $(\mathcal{D}, I)$ has two pieces of information: the average accuracy of criterion I on dataset $\mathcal{D}$ and the number of criteria with accuracy statistically lower than that of I on dataset $\mathcal{D}$. The statistical test used for criteria comparison is a one-tailed paired t-student test with a 95% confidence level. In general there was a balance between Twoing and GLχ^2, with GLSG being slightly worse. Running the same experiment using maximum depth 5 instead of 16 showed more balanced results. This suggests that our criteria should also be useful in boosting tree methods.

The results of Table 5.3.(a) also provide evidence of the potential of considering aggregated attributes. The accuracy obtained for the extended versions of datasets `Nursery`, `Cars` and `CoverType` are considerably higher than those obtained for the original versions. For `Contracep`, the effect is not clear.

Table 5.3.(b) presents the comparison between our methods and Twoing in another scenario, where we use c_{quad}, one of the bias-free criterion proposed in **?**, to select the attribute at each node of the tree. Then, both Twoing and our methods are used only for splitting the chosen attribute, which allows for a more direct comparison of their splitting ability. Again, we observed a balance between Twoing and GLχ^2. This experiment also showed that it is not possible to run c_{quad} in reasonable time for the `Shelter-15` dataset. This happens because it calculates a pseudoinverse of a matrix whose dimension grows with the number of values and classes, which is infeasible for large n and k.

Another key aspect to discuss is the computational cost of the proposed criteria. Table 5.10 shows the running time of Twoing, GLχ^2 and GLSG when they are used for both selecting and splitting purposes (the experiment of Table 5.3.(a)). When the number of classes is small all the criteria have very similar execution time, with Twoing being faster only on the `KDD98-2` dataset. As the number of classes increases, the GL-based methods become much faster than Twoing, with the turning point being around $k = 7$. For datasets with 15 classes our criteria are 30-300 times faster. We also ran experiments using all the classes available in both the `S.F. Crime` and `Shelter` datasets (39 and 22,

Table 5.3: Average accuracy and statistical tests for decision trees with depth at most 16 using only nominal attributes. The best accuracy for each dataset is bold-faced, even when multiple criteria have the same accuracy in the table because of rounding. The average and sum for the conditional inference tree criteria do not include the Shelter dataset, since the criteria don't run in reasonable time.

.6

Table 5.4: Attribute selection via Twoing and GL criteria

Dataset	Twoing		GLSG		GLχ^2	
`Adult`	**82.52**	**(2)**	82.38	(0)	82.43	(0)
`Mushroom`	**100**	**(1)**	100	(0)	99.99	(0)
`KDD98-2`	79.41	(0)	**80.65**	**(2)**	79.73	(0)
`KDD98-9`	38.54	(1)	37.97	(0)	**39.68**	**(2)**
`Nursery`	93.51	(1)	92.39	(0)	**93.74**	**(2)**
`Nursery-ext`	95.83	(1)	94.79	(0)	**96.02**	**(2)**
`Cars`	92.82	**(1)**	90.51	(0)	**92.88**	**(1)**
`Cars-ext`	**96.49**	**(2)**	90.89	(0)	94.44	(1)
`Contracep`	43.5	(0)	43.83	(0)	**43.85**	**(1)**
`Contracep-ext`	43.15	(0)	**44.32**	**(1)**	43.72	(0)
`CoverType`	64.09	(1)	**64.59**	**(2)**	63.61	(0)
`CoverType-ext`	**65.08**	**(1)**	**65.08**	**(1)**	65.05	(0)
`Poker`	**52.24**	**(2)**	49.88	(0)	51.83	(1)
`Shelter-15`	47.64	(1)	46.52	(0)	**48.09**	**(2)**
`S.F. Crime-15`	22.08	**(1)**	22.06	(0)	**22.08**	**(1)**
`Phonemes`	**37.13**	**(2)**	35.9	(0)	35.75	(0)
Average (Sum)	65.87	(17)	65.11	(6)	65.81	(13)

.4

Table 5.5: Attribute selection via conditional inference

CI-Twoing		CI-GLSG		CI-GLχ^2	
82.33	**(1)**	82.18	(0)	**82.35**	**(1)**
99.55	(0)	99.6	(0)	**99.64**	**(1)**
79.92	(0)	**81.53**	**(2)**	80.65	(1)
39.44	(0)	**40.65**	**(2)**	40.01	(1)
93.55	**(1)**	92.22	(0)	**93.56**	**(1)**
93.64	**(1)**	92.32	(0)	**93.65**	**(1)**
92.74	(1)	87.19	(0)	**92.92**	**(2)**
93.12	(1)	87.38	(0)	**93.34**	**(2)**
43.82	(0)	**44.05**	(0)	43.81	(0)
43.21	(0)	**43.78**	**(1)**	43.72	**(1)**
61.88	(0)	61.88	(0)	61.88	(0)
61.88	(0)	61.88	(0)	61.88	(0)
51.4	**(2)**	50.67	(0)	50.84	**(1)**
—		—		—	
22.07	(0)	22.07	(0)	**22.07**	**(1)**
33.91	**(2)**	31.26	(0)	33.32	(1)
66.16	(9)	65.24	(5)	66.24	(14)

Table 5.6: Average time in seconds of a 3-fold cross validation for building decision trees with depth at most 16. The fastest method for each dataset is bold-faced.

Dataset	k	Twoing	GLSG	GLχ^2
`Adult`	2	4.3	**4.3**	7.1
`Mushroom`	2	0.7	**0.6**	0.9
`KDD98-2`	2	**10.8**	57.8	60.8
`Contracep`	3	0.2	0.2	**0.1**
`Contracep-Ext`	3	**0.2**	0.3	0.3
`Cars`	4	0.3	0.2	**0.2**
`Cars-Ext`	4	**0.3**	0.4	0.4
`Nursery`	5	1.6	**1.4**	1.4
`Nursery-Ext`	5	**1.7**	3.9	3.6
`CoverType`	7	846.8	**373.4**	969.6
`CoverType-Ext`	7	338.6	**280.5**	505.7
`KDD98-9`	9	209.9	119.9	**112.5**
`Poker`	10	10.7	3.9	**3.7**
`Shelter-15`	15	5183.3	**155**	165.7
`S.F. Crime-15`	15	2667.9	94.2	**79.6**
`Phonemes`	15	3738.6	**8.7**	10.2

respectively). Twoing can not be executed in a reasonable time with that many classes, while GLSG and GLχ^2 ran in approximately 100 seconds on the `S.F. Crime` dataset and 300 seconds on the `Shelter` dataset. This behavior for the Twoing criterion is not surprising, since its running time has an exponential dependence of the number of classes k. Nonetheless, since the execution time for our criteria in this experiment grew in an approximately linear fashion with k, it suggests that they can also be used with datasets that have a much larger number of classes. It is also interesting to note that the aggregated attributes usually appeared at or near the root of the decision trees.

Table 5.11 shows experiments similar to those presented at Table 5.3.(a), but now using also the numeric attributes. We observed a significant gain in terms of accuracy for almost all datasets. The performance of GLχ^2 was competitive with that of Twoing for all datasets but `KDD98-9` and `CoverType`, where it was considerably better and worse, respectively.

5.3.1 Depth 5

In Table 5.8 we can see a balance among the different methods. In this experiment we also note that GLSG is more competitive with Twoing and GLχ^2 than in the experiment with depth 16 described in the paper. This suggests that GLSG loses competitiveness as the tree depth increases. Another interesting observation is that the GW-based criteria were about equal or

Table 5.7: Average accuracy and statistical test results for Decision Trees using both nominal and numeric attributes with depth at most 16.

Dataset	Twoing		GLSG		GLχ^2	
`Adult`	83	(1)	77.34	(0)	**83.21**	**(2)**
`KDD98-2`	**76.67**	**(2)**	76.36	(0)	76.04	(1)
`KDD98-9`	37.73	(1)	37.49	(0)	**43.44**	**(2)**
`Contracep`	**48.78**	**(1)**	48.01	(0)	48.66	**(1)**
`Contracep-Ext`	48.53	(0)	48.15	(0)	**48.6**	(0)
`CoverType`	85.14	(1)	**90.32**	**(2)**	81.38	(0)
`CoverType-Ext`	89.1	(1)	**92.03**	**(2)**	82.46	(0)
`Shelter-15`	53.79	(1)	52	(0)	**54.4**	**(2)**
`S.F. Crime-15`	26.66	(0)	26.71	(1)	**27.13**	**(2)**
Average (Sum)	61.04	(8)	60.93	(5)	60.59	(10)

Table 5.8: Average accuracy and statistical tests for decision trees with depth at most 5 using only nominal attributes. The best accuracy for each dataset is bold-faced.

Dataset	Twoing		GWSG		GWχ^2		GLSG		GLχ^2	
`Adult`	82.21	**(2)**	81.91	(1)	**82.24**	**(2)**	81.83	(0)	**82.24**	**(2)**
`Mushroom`	**100**	**(2)**	99.99	(0)	99.98	(0)	**100**	**(2)**	99.99	(0)
`KDD98-2`	80.47	(0)	81	(2)	80.74	(1)	**81.16**	**(3)**	80.51	(0)
`KDD98-9`	40.35	(2)	38.13	(0)	40.93	**(3)**	38.15	(0)	**41**	**(3)**
`Nursery`	88.25	**(3)**	88.03	(0)	88.2	(0)	**88.33**	**(3)**	88.2	(0)
`Nursery-Ext`	**93.82**	**(4)**	90.95	(0)	93.02	(2)	90.75	(0)	93.13	(2)
`Cars`	86.53	(1)	85.39	(0)	86.37	(1)	**87.93**	**(4)**	86.42	(1)
`Cars-Ext`	90.3	(0)	90.82	(1)	91.57	(3)	90.84	(1)	**91.9**	**(4)**
`Contracep`	43.77	(0)	43.96	(0)	**44.04**	(0)	43.89	(0)	44	(0)
`Contracep-Ext`	43.17	(0)	44.32	**(3)**	43.44	(0)	**44.35**	**(3)**	43.7	(0)
`CoverType`	52.97	(2)	**55.05**	**(3)**	51.07	(0)	**55.05**	**(3)**	51.07	(0)
`CoverType-Ext`	**64.48**	**(4)**	64.12	(2)	57.73	(0)	64.23	(3)	59.95	(1)
`Poker`	51.9	(2)	50.28	(1)	51.77	(2)	49.94	(0)	**51.91**	**(3)**
Average (Sum)	**68.67**	**(24)**	68.30	(13)	68.27	(20)	68.46	(22)	68.47	(22)

slightly inferior to their GL-based counterparts. This advantage is likely related with the fact that the weights of the cuts computed by the GL approach in this experiment are, in general, larger than those obtained by the GW algorithm.

For the experiment with the bias-free criteria, shown in table 5.9 we once again observed a balance among the different criteria, with a slight advantage towards our methods. Perhaps surprisingly, the bias free approach had significantly worse results for the datasets with extended attributes.

Table 5.10 shows the running time of each criterion when used for both selecting and splitting purposes. Twoing is the fastest method when the number of classes is small and the GL-based methods become competitive and even the fastest when the number of classes gets larger, as illustrated by the results on `KDD98` dataset. We also observe that the running time for the GW-based

Table 5.9: Average accuracy and statistical tests for Conditional Inference trees with depth at most 5 using only nominal attributes. The best accuracy for each dataset is bold-faced.

Dataset	CI-Twoing		CI-GLSG		CI-GLχ^2	
`Adult`	**81.96**	**(2)**	81.61	(0)	81.77	(1)
`Mushroom`	86.97	(0)	**94.79**	**(2)**	90.15	(1)
`KDD98-2`	81.29	(0)	**82.34**	**(2)**	81.68	(1)
`KDD98-9`	41.84	(0)	42	(0)	**42.26**	**(1)**
`Nursery`	**88.48**	**(1)**	88.3	(0)	88.47	**(1)**
`Nursery-Ext`	**88.48**	**(1)**	88.26	(0)	88.47	**(1)**
`Cars`	86.51	**(1)**	85.02	(0)	**86.57**	**(1)**
`Cars-Ext`	88.27	**(1)**	88.27	(0)	**88.32**	**(1)**
`Contracep`	43.83	(0)	**44.12**	**(1)**	43.87	(0)
`Contracep-Ext`	43.39	(0)	**43.81**	**(1)**	43.76	**(1)**
`CoverType`	54.1	(0)	54.1	(0)	54.1	(0)
`CoverType-Ext`	54.1	(0)	54.1	(0)	54.1	(0)
`Poker`	**51.05**	**(1)**	50.56	(0)	50.91	**(1)**
Average (Sum)	66.98	(7)	67.51	(10)	67.25	(10)

criteria were usually one or two orders of magnitude larger than the others. The only clear exception was in the `CoverType` dataset, where the number of samples is very large while the attributes' number of values is much smaller.

Finally, Table 5.11 shows experiments similar to those presented at Table 5.8, but now using also the numeric attributes. We observed a significant gain in terms of accuracy for all datasets except for KDD98-2. Also note that GLSG was inferior to both Twoing and GLχ^2.

Table 5.10: Average time in seconds of a 3-fold cross validation for building decision trees with depth at most 5. The fastest method for each dataset is bold faced.

Dataset	k	Twoing	GWSG	GWχ^2	GLSG	GLχ^2
Adult	2	**2.7**	41	88	3	4
Mushroom	2	**0.6**	6.9	8.6	0.9	1
KDD98-2	2	**4**	2162	3579	44	44
Contracep	3	**0.1**	0.6	1	0.1	0.1
Contracep-Ext	3	**0.1**	3.3	13	0.2	0.2
Cars	4	**0.1**	2.5	2.4	0.1	0.1
Cars-Ext	4	**0.2**	7.7	11	0.3	0.3
Nursery	5	**0.8**	4.7	5	1	0.9
Nursery-Ext	5	**1.1**	76	148	3.3	2.6
CoverType	7	349	265	**179**	245	308
CoverType-Ext	7	213	341	213	**182**	296
KDD98-9	9	132	3410	5899	97	**74**
Poker	10	7.4	6.5	11.2	2.2	**2.1**

Table 5.11: Average accuracy and statistical test results for Decision Trees using both nominal and numeric attributes.

Dataset	Twoing		GLSG		GLχ^2	
Adult	**84.2**	**(1)**	82.4	(0)	**84.2**	**(1)**
KDD98-2	80.9	(0)	**81.9**	**(1)**	**81.9**	**(1)**
KDD98-9	45.7	(1)	45.3	(0)	**48.7**	**(2)**
Contracep	**54.1**	**(1)**	52.9	(0)	54	**(1)**
Contracep-Ext	51.7	(0)	52.3	(0)	**53.3**	**(2)**
CoverType	**70.3**	**(2)**	68.5	(0)	69.2	(1)
CoverType-Ext	**70.9**	**(2)**	70.3	(1)	69.2	(0)
Average (Sum)	64.64	(9)	63.98	(2)	64.99	(11)

6
Conclusions

In this paper we proposed a framework for designing splitting criteria for handling multi-valued nominal attributes. Criteria derived from our framework can be implemented to run in polynomial time in n and k, with theoretical guarantee of producing a split that is close to the optimal one.

Experiments over 11 datasets suggest that the GLχ^2 criterion, obtained from our framework, is competitive with the well-established Twoing criterion in terms of both accuracy and speed for datasets with a small number of classes ($k \leq 7$). It is also much faster than Twoing when the number of classes is greater than 10, while keeping a comparable accuracy.

Therefore, our methods are an interesting alternative to deal with datasets with a large number of classes that contain nominal attributes with a large number of different values, since those cannot be properly handled by Twoing due to its exponential running time dependence on the number of classes.

Furthermore, our experiments also reinforce the potential of aggregating attributes as a tool for improving the accuracy of decision trees. An interesting topic for future research is evaluating the behavior of our criteria in boosted tree methods. Another direction for future work is developing new methods for automatic aggregating attributes, or improving the available ones.

Bibliography

Austin Animal Center (2016), 'Shelter animal outcomes dataset', *URL: kaggle.com/c/shelter-animal-outcomes*.