Reversing the Classification of Weighted Rule Ensembles

Tibérius O. Bonates
Federal University of Ceará
Fortaleza, CE, Brazil
tb@ufc.br

José W.V. Morais Neto
Federal University of the Semi-Arid
Mossoró, RN, Brazil
moraisneto_@hotmail.com

Abstract

We introduce the problem of classification reversal of weighted rule ensembles, i.e., classifiers consisting of a set of weighted decision rules. Given a set of decision rules R, built for a two-class classification task on a binary data set, and a non-negative real weight $w(r)$ for each rule $r \in R$, the weighted rule ensemble classifier $F = (R, w)$ classifies an observation $\mathbf{x}$ according to a linear function of the weights of the rules satisfied by $\mathbf{x}$. The problem of classification reversal of F on observation $\mathbf{x}$ can be briefly described as finding a smallest set of features of $\mathbf{x}$ such that their simultaneous modification is enough to bring about the reversal of the classification of $\mathbf{x}$ by F. We show the problem to be $\mathcal{NP}$-hard, describe an optimization model for finding such a set of features and discuss variants of the model. Our preliminary computational tests based on publicly available data suggest that the model can be solved for data sets of moderate size, warranting further investigation regarding its application to real-life data sets.

1 Introduction

Let $D = \{(\mathbf{x}^1, y^1), (\mathbf{x}^2, y^2), \ldots, (\mathbf{x}^p, y^p)\}$ be a data set, where each $\mathbf{x}^i \in \{0,1\}^n$ is a binary n-vector corresponding to an observation and $y^i \in \{-1, +1\}$ corresponds to the class to which $\mathbf{x}^i$ belongs. We shall refer to the features of D, i.e., the coordinates of the observations in D, as $X_1, X_2, \ldots, X_n$.

A *decision rule* (or, simply, a *rule*) consists of a set of tests involving a subset of the features. Any test in a rule corresponds to checking whether the value of a feature X_i equals a value $v \in \{0,1\}$. In other words, a decision rule r is a conjunction of tests of the type $(X_i = v_i)$, $i \in A \subseteq \{1, \ldots, n\}$, $v_i \in \{0,1\}, \forall i$. Given an observation $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, we shall write $r(\mathbf{x}) = \bigwedge_{i \in A}(x_i = v_i)$ and say that r is satisfied by $\mathbf{x}$ if and only if the value of the Boolean conjunction $r(\mathbf{x})$ is true.

Consider a set R of decision rules induced from D. We shall say that a rule r is *active* on $\mathbf{x}$ if r satisfied by $\mathbf{x}$, i.e., if $r(\mathbf{x})$ is true. By assuming that decision rule r of R is associated with exactly one of the classes, we can define $\text{sign}(r)$ as follows: $\text{sign}(r) = -1$ if r is associated with class -1, and $\text{sign}(r) = +1$ if r is associated with class $+1$. We shall also say that a rule r is positive (negative) if $\text{sign}(r) = +1$ (respectively, $\text{sign}(r) = -1$). Thus, each rule will provide, by itself, a classification decision regarding any given observation.

In rule ensemble classifiers, such as LAD (Boros et al. 1997; 2000) and Random Forests (Breiman 2001; Ho 1998), multiple decision rules are evaluated in order to classifiy a single observation. Therefore, there might be conflicting decisions of the members of the ensemble regarding the very same observation. A commonly used way of combining the decisions of a set R of rules on the classification of an observation $\mathbf{x}$ is to take the majority vote among them: $\mathbf{x}$ is classified according to the class with the largest number of classification decisions in its favor (Breiman 2001).

A more general way of reaching such a combined decision in two-class classification tasks is to define a weight function $w : R \to \mathbb{R}_+$ and to consider the sum of the weights of the rules satisfied by $\mathbf{x}$ (see, e.g., (Boros et al. 2000)). Let us define $\Delta(R, w, \mathbf{x})$ as the difference between the sum of the weights of the positive rules that are active on $\mathbf{x}$ and the sum of the weights of the negative rules that are active on $\mathbf{x}$, given by the pseudo-Boolean function:

$$\Delta(R, w, \mathbf{x}) = \sum_{r \in R} w(r)\ \text{sign}(r)\ r(\mathbf{x}). \tag{1}$$

Whenever the rule set R and the weight function w are clear from the context, we shall simply write $\Delta(\mathbf{x})$ to denote $\Delta(R, w, \mathbf{x})$. If $\Delta(\mathbf{x}) \neq 0$, then observation $\mathbf{x}$ is classified as belonging to class $\text{sign}(\Delta(\mathbf{x}))$; otherwise, the observation is not classified by F.

We shall refer to a set of decision rules R, trained on a two-class data set of binary vectors, along with a weight function $w : R \to \mathbb{R}_+$, as a *weighted rule ensemble* $F = (R, w)$. From now on we shall refer to the classification decision of such an ensemble F on an observation $\mathbf{x}$ as $F(\mathbf{x})$, and define it as

$$F(\mathbf{x}) = \begin{cases} \text{sign}(\Delta(\mathbf{x})), & \text{if } \Delta(\mathbf{x}) \neq 0; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In this paper, we shall be concerned with the problem of, given a weighted rule ensemble F and an observation $\mathbf{x}$, with $\Delta(\mathbf{x}) \neq 0$, selecting a minimum-size set $S \subseteq \{1, \ldots, n\}$ of feature indices of $\mathbf{x}$ such that if the values of the features indexed by S are reversed in $\mathbf{x}$, then the sign of

$F(\mathbf{x})$ is also reversed. Formally, we are interested in finding a smallest set S such that if we define an observation $\mathbf{x}' \in \{0,1\}^n$ as

$$\mathbf{x'}_i = \begin{cases} 1 - \mathbf{x}_i, & \text{if } i \in S, \\ \mathbf{x}_i, & \text{otherwise,} \end{cases} \quad (3)$$

then we have $F(\mathbf{x}) \neq F(\mathbf{x}')$.

The remainder of the paper is organized as follows. In Sections 2 and 3 we discuss the problem of classification reversal and review previous related work. In Section 4 we show that the classification reversal of a weighted rule ensemble is $\mathcal{NP}$-complete. In Sections 5 and 6 we introduce a binary optimization model for the problem and discuss some of its variants. Preliminary computational results are reported in Section 7 for the case of a majority vote classifier. Finally, Section 8 concludes the paper outlining the results and pointing out directions for future work.

2 Classification Reversal

As with many other classification methods, there is no simple way of determining the relative influence of individual features in the final classification of a given observation by a decision rule ensemble, in general. Indeed, such classifiers are typically composed of a number of rules, each possibly involving a different subset of features, combined in different ways (Ho 1998). Any individual feature might appear in several rules and participate in more than one type of test.

For instance, a random forest (RF) classifier built on a data set containing gene expression levels of cancer patients and controls will typically consist of hundreds decision trees. Each tree can be understood as a set of decision rules; a decision forest consists of a weighted rule ensemble, in which each rule is assigned a weight equal to 1. Each test in such a RF classifier verifies whether the expression level of a specific gene is above or below a certain threshold (see, e.g., (Statnikov and Aliferis 2007)). A gene that is highly influential for the classification decision of such a decision forest can be considered a *biomarker*, i.e., a biochemical characteristic of an organism that is associated with the presence or severity of a medical condition. Discovering a biomarker – or sets of genes which, collectively, act as a biomarker – associated with a specific disease can lead to improved methods for detecting such a condition and to novel treatment options (Chin et al. 2011).

A particular form of therapy derived from the knowledge of genomic information is called gene therapy and involves the delivery of genetic material to an individual's cells, in some cases with the aim of replacing modified genes that are involved in the underlying mechanism of a certain type of cancer. The development of personalized gene therapies based on genomic data analyses is intrinsically linked to the identification of biomarkers, and constitutes a promising area in cancer research, as suggested in (Chin et al. 2011): "it [is] increasingly clear that the ability to perform prospective and comprehensive molecular profiling of tumors will (...) enable genome-informed personalized cancer medicine".

In view of this, a natural question concerning the importance of features is: "Given an observation $\mathbf{x} = (x_1, \ldots, x_n)$ and a weighted rule ensemble classifier F, with rule set R and weight function w, is there a single feature X_j with the property that if we change the value of X_j in $\mathbf{x}$, then we also change the classification given by $F(\mathbf{x})$?" In other words, is there j such that if we define $\mathbf{x}' = (x_1, \ldots, x_{j-1}, 1 - x_j, x_{j+1}, \ldots, x_n)$, then we have $F(\mathbf{x}) \neq F(\mathbf{x}')$? This is a simple question that can be answered in polynomial time by inspection.

In general, however, there is no such feature X_j. Therefore, we consider a modified version of the question in which we ask for a minimum-size subset $S \subseteq \{1, \ldots, n\}$ of feature indices such that, if the values of the features indexed by S are all simultaneously switched in $\mathbf{x}$, then the sign of $F(\mathbf{x})$ is changed. Since we are dealing with a binary classification task, we shall call the operation of changing the classification of an observation $\mathbf{x}$ "reversing the classification of $\mathbf{x}$ by F."

Clearly, $F(\mathbf{x}) \neq F(\mathbf{x}')$ implies that modifying the components of $\mathbf{x}$ specified by S causes some of the rules in R to classify the modified observation $\mathbf{x}'$ differently from how they classify $\mathbf{x}$. For each rule $r \in R$, we either have $r(\mathbf{x}') = r(\mathbf{x})$ or $r(\mathbf{x}') \neq r(\mathbf{x})$.

The scenario $r(\mathbf{x}') = r(\mathbf{x})$ corresponds to rules that classify $\mathbf{x}'$ the same way they classify $\mathbf{x}$, i.e., the changes in the S-features do not affect the classification decision of such rules. On the other hand, the $r(\mathbf{x}') \neq r(\mathbf{x})$ scenario corresponds to those rules whose classification are affected by the changes in S. It is the net effect of occurrences of this second scenario on the value of $\Delta(\mathbf{x})$ that can realize the intended classification reversal.

3 Review of Previous Work

In (Axelrod et al. 2004) a LAD classifier – i.e., a classifier based on a weighted set of decision rules (Boros et al. 1997) – was used for building a proof of concept of the idea of modifying a given observation's classification by means of feature changes. In that work the set of rules was derived by a specialized algorithm (Alexe and Hammer 2006) and a linear program was used to determine the rule weights (Boros et al. 2000).

An optimization model was developed in (Axelrod et al. 2004) for the purpose of finding a set of feature changes that minimized a so-called discriminant function $\delta : \mathbb{R}^n \to \mathbb{R}$, similar to the Δ function introduced in Section 1, but defined over $\mathbb{R}^n$. Assuming that $\delta(\mathbf{x}) > 0$, the objective was to obtain from $\mathbf{x}$ a modified observation $\mathbf{x}'$ satisfying $\delta(\mathbf{x}') < 0$ and having $|\delta(\mathbf{x}')|$ as large as possible. The model included constraints that retained the active/inactive status of certain rules: each negative rule r that was active on $\mathbf{x}$ should remain active on $\mathbf{x}'$. Conversely, no positive rule r that was inactive on $\mathbf{x}$ should be active on $\mathbf{x}'$. In other words, one was only allowed to activate rules of negative weight and/or deactivate positively weighted rules.

Moreover, since in (Axelrod et al. 2004) features were assumed to be real-valued, each feature of a given observation could have its value modified in a number of different ways: changes could vary in sign (i.e., increase/decrease) and absolute value. A natural way of comparing two sets A and B of feature changes was to verify if the following

conditions held simultaneously: (i) the modified features in A were a subset of those in B; (ii) the modifications in A had the same sign as their counterparts in B; and (iii) the modifications in A were at most as large (in absolute value) as their counterparts in B. A relation of this type between two sets of feature changes indicated that one of them (set B) was at least as demanding as the other (set A) in terms of the amount of changes effected.

Since an integer linear programming model was used to obtain an optimal set of features, there was no straightforward way of generating all alternative solutions to the model. Thus, if one was interested in comparing alternative solutions (as described above) a substantial amount of manual intervention or additional software was required.

The main differences from the problem discussed in this paper to the one described in (Axelrod et al. 2004) are that: (i) the current problem does not involve constraints on the activity status of rules; and (ii) the current problem does not take into account rule weights or magnitudes of feature changes. Thus, the problem introduced here is neither a generalization nor a particular case of the one discussed in (Axelrod et al. 2004).

4 Complexity of Reversing the Classification of a Weighted Rule Ensemble

In this section we show that the problem of reversing the classification of a weighted rule ensemble, as described above, is $\mathcal{NP}$-complete.

Firstly, we formally define the decision version of classification reversal the problem. In our definition, a binary test in a decision rule is a test that takes either of the two forms: $X_j = 0$ or $X_j = 1$, for some $j \in \{1, \ldots, n\}$. Moreover, let $\mathbf{x} \in \{0,1\}^n$ be an observation and let $S \subseteq \{1, \ldots, n\}$. We shall denote by $\mathbf{x}^S$ the observation obtained by switching the values of the components of $\mathbf{x}$ that are indexed by S, as shown in Equation 3.

RULE ENSEMBLE CLASSIFICATION REVERSAL
INPUT: A weighted rule ensemble F, with rule set R and weight function $w : R \to \mathbb{R}_+$, in which each rule involves only binary tests, an observation $\mathbf{x} \in \{0,1\}^n$, and a positive integer k.
QUESTION: Is there a set $S \subset \{1, \ldots, n\}$, $|S| \leq k$, such that $F(\mathbf{x}^S) \neq F(\mathbf{x})$?

We now prove the $\mathcal{NP}$-completeness of RULE ENSEMBLE CLASSIFICATION REVERSAL (RECR) by means of a polynomial-time reduction from HITTING SET, whose definition we review below.

HITTING-SET
INPUT: A collection $\mathcal{C}$ of subsets of a finite set E and a positive integer d.
QUESTION: Is there a subset $A \subseteq E$, $|A| \leq d$, such that A contains at least one element from every member of $\mathcal{C}$.

Before proving the main result, we define a construction that extends a given rule into a decision tree.

Definition 4.1 (Extended Decision Tree) *Given a rule r involving a subset of the binary features $\{X_1, \ldots, X_n\}$, we shall build a decision tree based on r as follows. Initially, let us create a tree $T(r)$ containing a single node, which takes on the simultaneous roles of root and only leaf of $T(r)$. Then, we inspect the tests in r according to an arbitrary but fixed order. For each test a branch is added to the current leaf t of $T(r)$, thereby creating a new node, which becomes the only leaf of $T(r)$. We proceed in this fashion until all tests in r have been processed. At this point, we associate the only leaf of tree $T(r)$ with the class of r. Finally, for each non-leaf node we add the missing branch, thus creating new leaves, which are then associated with the opposite class of r. Figure 1 illustrates the construction of such a tree from the negative rule $(X_2 = 0, X_3 = 1, X_5 = 1)$. The white leaf is associated with the negative class, while the black leaves are associated with the positive class.*

Since each path from the root of a decision tree to one of its leaves coincides with a decision rule, and the set of rules obtained from such paths provides the same classification as the tree itself, we shall identify each tree $T(r)$ with the set of rules corresponding to it. It is important to remark that, due to the inherent structure of a decision tree, exactly one of the rules in $T(r)$ is satisfied by any given observation.

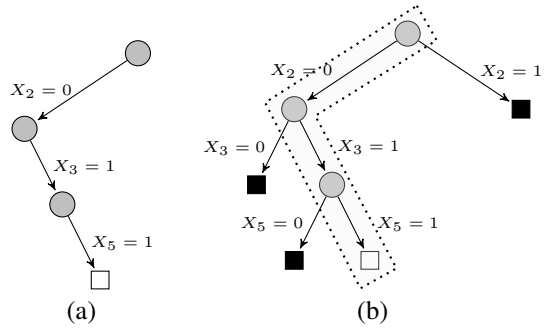

Figure 1: (a) Intermediate step in extending a decision tree from negative rule $r = (X_2 = 0, X_3 = 1, X_5 = 1)$. Only the branches corresponding to tests in r are shown; (b) Fully extended decision tree built from rule $r = (X_2 = 0, X_3 = 1, X_5 = 1)$. The path corresponding to r is highlighted. The entire set of decision rules corresponding to the tree is $T(r) = \{r, (X_2 = 1), (X_2 = 0, X_3 = 0), (X_2 = 0, X_3 = 1, X_5 = 0)\}$.

It is important to note that the Extended Decision Tree corresponding to a rule r can be created in time that is linear on the number of tests in r.

Theorem 4.2 RULE ENSEMBLE CLASSIFICATION REVERSAL *is $\mathcal{NP}$-complete.*

Proof First, let us note that RECR is in $\mathcal{NP}$. Consider an instance of RECR consisting of a weighted rule ensemble $F = (R, w)$, with R being a set of rules and w being a weight function $w : R \to \mathbb{R}_+$, an observation $\mathbf{x} \in \{0,1\}^n$ and a positive integer k. We can verify this instance in polynomial time by inspecting a solution P to it: the modified observation can be computed in time that is linear on n; it is enough to verify the classification decision by each rule $r \in R$ on both the original and the modified observations

and to compute the values of $\Delta(\mathbf{x})$ and $\Delta(\mathbf{x}')$. Each rule contains up to n tests. Thus, the entire process can be carried out in $\mathcal{O}(|R|\ n)$-time.

Second, we consider the $\mathcal{NP}$-hardness of RECR. Let $I_H = (E, \mathcal{C}, d)$ be an instance of HITTING-SET with a n-element ground set $E = \{e_1, \ldots, e_n\}$, a collection $\mathcal{C} = \{C_1, \ldots, C_m\}$ of subsets of E, and an upper bound d on the cardinality of a solution.

Let us define an observation $\mathbf{x} = (1, \ldots, 1) \in \{0,1\}^{n+m}$ and associate with each member C_i of $\mathcal{C}$ a negative decision rule R_i, which consists exclusively of tests of the type $X_j = 1$, for all elements $e_j \in C_i$. We shall denote by $T(R_i)$ the set of rules corresponding to the Extended Decision Tree of R_i (Definition 4.1).

Let us also introduce two additional sets of simple rules, each of which consists of a single test:

$$P_i = (X_{n+i} = 0), i = 1, \ldots, m$$
$$N_i = (X_{n+i} = 1), i = 1, \ldots, m,$$

with each rule P_i being positive and each rule N_i being negative.

Let us now define $R = \{T(R_i) : i = 1, \ldots, m\} \cup \{P_i : i = 1, \ldots, m\} \cup \{N_i : i = 1, \ldots, m\}$ as the multiset of rules defined previously. We define R as a multiset because we will be interested in taking into account the weight of each individual rule, regardless of the fact that two or more copies of a rule might appear in the sets that compose R. Moreover, let us define $w : R \to \mathbb{R}_+$ such that $w(r) = 1$, for every $r \in R$. (Note that this particular weighting scheme is equivalent to using the majority vote among rules.) Now, we can define a weighted rule ensemble $F = (R, w)$. By construction, we have that: (i) the only negative rule from set $T(R_i)$ is active on $\mathbf{x}$, for each $i = 1, \ldots, m$; (ii) every positive rule P_i $(i = 1, \ldots, m)$ is inactive on $\mathbf{x}$; and (iii) every negative rule N_i $(i = 1, \ldots, m)$ is active on $\mathbf{x}$. Therefore, $\Delta(\mathbf{x}) = -2m$, implying $F(\mathbf{x}) = -1$.

The families of simple rules P_i and N_i defined above can be generated in polynomial time on m. Furthermore, each extended decision tree $T(R_i)$ can be generated in time proportional to n. Thus, then the entire weighted rule ensemble F can be generated in polynomial time on n and m. By defining an instance $I_R = (F, \mathbf{x}, k)$ of RECR, with $k = d+1$ being an upper bound on the number of features modified in a solution to I_R, we have effectively exhibited a polynomial-time procedure to transform a Hitting Set instance I_H into an instance of RECR.

Next, we show that I_H is a 'yes' instance if and only if the corresponding I_R instance is also a 'yes' one.

Any solution A to I_H (with $|A| \leq d$) can be transformed into a solution B to I_R, with $|B| \leq k$, by simply augmenting A with an arbitrary element from the set $Q = \{n+1, \ldots, n+m\}$. Indeed, A intersects every set C_i from $\mathcal{C}$, and, due to the structure of the rules R_i, the set of changes to $\mathbf{x}$ induced by A are enough to deactivate the m negative rules in the set $\{R_1, \ldots, R_m\}$, while activating one positive rule from each set $T(R_i)$, $i = 1, \ldots, m$. By modifying the value of an additional feature X_j, with $j \in Q$, we deactivate yet another negative rule (namely, N_{n+j}) and activate one positive rule (P_{n+j}), ensuring that the number of active positive rules exceeds the number of active negative rules. Thus, the set $B = A \cup \{j\}$, for any $j \in Q$, induces the reversal of the classification of the modified observation $\mathbf{x}^B$ by F, using a total of $|B| = |A| + 1 \leq k$ feature changes.

We now show how a set $P \subseteq \{1, \ldots, n+m\}$, with $|P| \leq k = d+1$, which is a solution to I_R, can be transformed into a solution to I_H, with cardinality at most $|P| - 1$. Note that, by construction of the rule multiset R, the solution P specifies changes to observation $\mathbf{x}$ which result in at least $m+1$ positive rules becoming active, what means that P must include at least one element from the set Q defined above. If $|P \cap Q| = 1$, then $P \setminus Q$ is clearly a hitting set for I_H, with cardinality at most d. If $|P \cap Q| > 1$, we claim that it is possible to carry out a polynomial-time transformation of P into a solution P' to I_R satisfying $|P' \cap Q| = 1$ and $|P'| \leq |P|$.

In order to accomplish that, let us first make sure that P is minimal with respect to the inclusion of elements from $P \cap Q$ (i.e., P does not include more elements from Q than necessary to reverse the classification of F). We can clearly enforce this in polynomial time, by keeping track of how many positive rules from the sets $T(R_1), \ldots, T(R_m)$ are satisfied by $\mathbf{x}^P$ and discarding (from P) as many elements that lie in the intersection $P \cap Q$ as possible, while preserving $F(\mathbf{x}) \neq F(\mathbf{x}^P)$.

If we still have $|P \cap Q| > 1$ after simplifying P via this procedure, then there must exist $|P \cap Q| - 1$ members of $\mathcal{C}$ that have no intersection with P (and, therefore, $|P \cap Q| - 1$ active negative rules in $\{T(R_1), \ldots, T(R_m)\}$). Then, there must exist a set Z of at most $|P \cap Q| - 1$ elements from $\{1, \ldots, n\} \setminus P$ such that every member of $\mathcal{C}$ is intersected by $P' = (P \cup Z) \setminus W$, for any set $W \subset P \cap Q$ of cardinality $|P \cap Q| - 1$. Indeed, the set Z can be constructed in polynomial time by selecting at most one element from each of the members of $\mathcal{C}$ that have no intersection with P. The set P' thus obtained is also a solution to I_R, since $|P'| \leq k$ and the net effect of removing elements from W on $\Delta(\mathbf{x}')$ is compensated by the introduction of the elements in Z, assuring $F(\mathbf{x}) \neq F(\mathbf{x}')$. Thus, we obtain a solution P' to I_R with $|P' \cap Q| = 1$, and $P' \setminus Q$ induces a solution of cardinality $|P'| - 1 \leq d$ to I_H.

The theorem follows. ∎

Since obtaining a solution to the optimization version of the problem allows us to solve its decision version, we have that the optimization version of RECR is $\mathcal{NP}$-hard.

5 An Optimization Model for Classification Reversal

We now introduce a binary optimization model for the optimization version of RECR.

Let us write the set of rules in the decision ensemble in question as $R = \{r_1, \ldots, r_{|R|}\}$. Each rule $r_j \in R$ will be regarded as a set of tests such as $r_j = \{(X_k = v_k) : v_k \in \{0,1\},\ k \in I(r_j)\}$, with $I(r_j)$ denoting the index set of the features involved in rule r_j. We shall consider $\mathbf{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$ to be the observation of interest.

Before introducing the model, let us define the decision variables involved. Variable $z_k \in \{0,1\}$ corresponds to the

decision of whether or not to modify the value of the k-th feature of observation $\mathbf{x}$, for $k = 1, \ldots, n$:

$$z_k = \begin{cases} 1, & \text{if the value of } x_k \text{ is to be modified;} \\ 0, & \text{otherwise.} \end{cases}$$

For each rule $r_j \in R$, a binary decision variable y_j is associated with the activity status of r_j on the modified observation: $y_j = 1$ if r_j is active, and $y_j = 0$ otherwise.

We must ensure that each rule r_j can only be active on the modified observation if the following conditions hold: (i) $z_k = 0$, whenever r_j contains the test $(X_k = x_k)$; and (ii) $z_k = 1$, whenever r_j contains the test $(X_k = 1 - x_k)$. In other words, we must ensure that

$$y_j = \prod_{k \in A(\mathbf{x}, r_j)} (1 - z_k) \prod_{k \in C(\mathbf{x}, r_j)} z_k,$$

where $A(\mathbf{x}, r_j) = \{k : (X_k = x_k) \in r_j\}$ and $C(\mathbf{x}, r_j) = \{k : (X_k = 1 - x_k) \in r_j\}$ are the index sets corresponding to the components of $\mathbf{x}$ which are in *agreement* (respectively, in *conflict*) with the tests in rule r_j. Enforcing conditions (i) and (ii) above can be accomplished by the introduction of two types of constraints. The first type forces y_j to zero if any of the conditions (i) or (ii) is violated:

$$|I(r_j)|\, y_j \leq |I(r_j)| - \sum_{k \in A(\mathbf{x}, r_j)} z_k - \sum_{k \in C(\mathbf{x}, r_j)} (1 - z_k), \quad (4)$$

Indeed, if any of the two summations in the right-hand side of (4) adds up to a nonzero value, then y_j is forced to be zero; otherwise, y_j is free to take either value 0 or 1. The second type of constraint forces the value of y_j to be 1 when rule r_j is active:

$$y_j \geq 1 - \sum_{k \in A(\mathbf{x}, r_j)} z_k - \sum_{k \in C(\mathbf{x}, r_j)} (1 - z_k). \quad (5)$$

Contrary to the case of constraint (4), if both summations in the right-hand side of (5) have zero value, then the constraint forces y_j to take the value 1. Clearly, constraints (4) and (5) are enough to assign the appropriate 0-1 value to each variable y_j, with $j = 1, \ldots, |R|$.

Alternatively, if the problem is being modeled as an integer linear program, a tighter formulation can be obtained by expressing constraints (4) in the following disaggregated form, as shown in (Crama and Hammer 2011):

$$\begin{aligned} y_j &\leq 1 - z_k, \quad \forall k \in A(\mathbf{x}, r_j) \\ y_j &\leq z_k, \quad \forall k \in C(\mathbf{x}, r_j). \end{aligned}$$

We shall refer to $\mathbf{x}'$ as the observation obtained after the feature changes prescribed by the z_k variables are applied to $\mathbf{x}$. Since we want to make sure that $F(\mathbf{x}) \neq F(\mathbf{x}')$, a constraint is needed to enforce that the set of active rules is changed in such a way that $\text{sign}(\Delta(\mathbf{x}')) \neq \text{sign}(\Delta(\mathbf{x}))$. The constraint depends on the value of $F(\mathbf{x})$ and on the value of a positive real-valued parameter U, which reflects a lower bound on the value of $|\Delta(\mathbf{x}')|$, and shall take the following form:

$$F(\mathbf{x}) \sum_{j=1}^{|R|} w(r_j)\, \text{sign}(r_j)\, y_j \leq -U. \quad (6)$$

Finally, the objective function is simply

$$\sum_{k=1}^{n} z_k. \quad (7)$$

Thus, we can formulate the optimization model for reversal of the classification of $\mathbf{x}$ by F as follows:

$$\begin{gathered} \text{(CR) minimize (7), subject to: (4)-(6),} \\ z_k \in \{0, 1\},\ k = 1, \ldots, n, \\ y_j \in \{0, 1\},\ j = 1, \ldots, |R|. \end{gathered}$$

An optimal solution to model (CR) clearly corresponds to a minimum-size set of feature changes that is enough to reverse the classification of $\mathbf{x}$ by F. The model can be formulated as an integer linear program or as a constraint programming model and solved via a standard solver. In the next section we show how the model can be modified in order to account for a secondary optimization criterion, as well as some practical side constraints.

6 Variants of the Basic Model

In this section, we discuss some variants of model (CR) and describe a partial order on its set of feasible solutions.

As previously discussed, we assume that the given observation $\mathbf{x}$ satisfies $\Delta(\mathbf{x}) \neq 0$. Thus, we can write $F(\mathbf{x}) = \text{sign}(\Delta(\mathbf{x}))$. In what follows, whenever we refer to $\mathbf{x}'$ we are actually referring to the modified observation that is obtained after the feature changes prescribed by a solution to (CR) are applied to $\mathbf{x}$.

The *margin* of an ensemble classifier F on an observation $\mathbf{x}$ corresponds to a concept of distance from x to the decision boundary defined by F. According to one of the usual definitions (see, e.g., (Schapire et al. 1998)), the margin of classifier F on observation $\mathbf{x}$ can be expressed as

$$m(F, \mathbf{x}) = \frac{\Delta(\mathbf{x})}{\sum_{r \in R} w(r)}. \quad (8)$$

The margin is commonly associated with the confidence level of the classification of $\mathbf{x}$ provided by F (Schapire et al. 1998). The larger the absolute value of $\Delta(\mathbf{x})$ the higher the confidence in the classification of $\mathbf{x}$ by F. Indeed, if we assume that each individual rule is a relatively accurate classifier on its own, then a large margin corresponds to a large body of evidence in favor of the classification of $\mathbf{x}$ by F. In our case, the denominator of (8) is constant. Thus, if we assume that $F(\mathbf{x}) \neq F(\mathbf{x}')$, we might simply use the absolute value of $\Delta(\mathbf{x}')$ as a measure of confidence on, or effectiveness of the reversal of the classification of $\mathbf{x}$ by F.

Consequently, as a secondary objective to the one of finding a minimum-size set of feature changes that realize the classification reversal, it is desirable that $|\Delta(\mathbf{x}')|$ be made as large as possible. In particular, given two or more optimal solutions to model (CR), it is reasonable to break the tie by selecting a solution that achieves the largest absolute value of $\Delta(\mathbf{x}')$. Model (CR) can be easily modified to account for this secondary objective (i.e., the number of feature changes being equal, select a solution whose value of

$|\Delta(\mathbf{x}')|$ is largest). First, we replace constraint (6) by the following two constraints:

$$v = \sum_{j=1}^{|R|} w(r_j)\ \text{sign}(r_j)\ y_j \quad (9)$$

$$F(\mathbf{x})\ v \leq -U, \quad (10)$$

where v is a continuous decision variable unconstrained in sign and whose value is $\Delta(\mathbf{x}')$ (see Equation (1)). Then, we replace objective function (7) by

$$\sum_{k=1}^{n} z_k - \frac{F(\mathbf{x})}{\sum_{r\in R} w(r)}\ v\ , \quad (11)$$

which is also to be minimized. Note that $F(\mathbf{x})v/\sum_{r\in R} w(r) \in (-1, 0)$; therefore, this penalty term does not interfere in the optimization of the original objective function (7).

Moreover, while the value of U can be adjusted in order to reflect a more adequate minimum confidence level on the classification $F(\mathbf{x}')$ (see Equation (6)), we can also impose an upper bound on the value of $|\Delta(\mathbf{x}')|$. Imposing such a bound might be a sensible choice, since a large value of $|\Delta(\mathbf{x}')|$ will likely require changing the values of several features, resulting in an unrealistic scenario for certain applications, such as the gene therapy scenario described in Section 2. Thus, one might replace constraint (10) by

$$-T \leq F(\mathbf{x})\ v \leq -U, \quad (12)$$

for some positive real value $T \geq U$.

In light of this discussion, it is possible to define a partial order relation between feasible solutions of an instance of model (CR) based on their sets of feature changes and their corresponding values of $\Delta(\mathbf{x}')$. Let us define E as the set of all pairs (S, V_S), where the set of feature indices S corresponds to a feasible solution to (CR), i.e., to the z_k variables at value 1, and $V_S = \Delta(\mathbf{x}^S)$ is the Δ-value corresponding to solution S. For any two given feasible solutions S_1 and S_2 to (CR), a natural order relation $\prec$ can be defined as

$$(S_1, V_{S_1}) \prec (S_2, V_{S_2}) \iff S_1 \subset S_2 \ \text{ and } \ |V_{S_1}| < |V_{S_2}|.$$

The pair $\mathbf{P} = (E, \prec)$ defines a strict partial order (Trotter 2001). Note that if $S_1 \subset S_2$ and $|V_{S_1}| = |V_{S_2}|$, then S_2 does not represent any actual gain with respect to S_1 (if anything, solution S_2 can be regarded as inefficient with respect to S_1, since it uses more features than S_1, but does not achieve a larger value of $|\Delta(\mathbf{x}')|$). This poset structure is similar to the one proposed in (Axelrod et al. 2004).

Among the minimal elements of $\mathbf{P}$ are the optimal solutions of (CR). The maximal elements of $\mathbf{P}$ are the solutions of (CR) satisfying the property that no other solution with a superset of feature changes achieves a larger absolute value of $\Delta(\mathbf{x}')$. A chain in $\mathbf{P}$ is a sequence of solutions to (CR) that defines a family of nested sets of features, along which the absolute value of $\Delta(\mathbf{x}')$ (and, therefore, the confidence level of the classification reversal) is non-decreasing.

By analyzing the structure of poset $\mathbf{P}$ one might be able to more accurately decide about which solutions are more adequate for implementation in a practical context, given: (i) the maximum acceptable number of feature changes; (ii) our ability/willingness to simultaneously carry out certain groups of feature changes; and (iii) the most appropriate ratio between $|\Delta(\mathbf{x}^S)|$ and the size of the set S.

7 Preliminary Experiments

In this section we present computational results based on the solution of model (CR) for a particular type of RECR instance derived from a random forest classifier. The set of rules was obtained from a decision forest generated from a binary data set: for every tree T in the forest, all paths from T's root to a leaf of T were enumerated, thus giving rise to a set of decision rules. All rules obtained in this way were assigned a weight equal to 1, which means that our classifier employs a majority vote rule for deciding how to classify a given observation.

The model was implemented in C++ with the use of the Gecode constraint programming (CP) library (Schulte, Tack, and Lagerkvist 2010). The reason for selection CP as a solution technique is two-fold. First, since the classifier uses a majority voting scheme, the decision variable v in constraints (9) and (10) is integer-valued and can be handled by a CP solver with support for variables with integer domains. Second, CP allows for the systematic enumeration of all feasible solutions of (CR) without an excessive computational overhead due to re-optimization. In fact, if we constrain the absolute value of $|\Delta(\mathbf{x}')|$ to be within a relatively small interval (using a constraint such as (12)), we might be able to construct the corresponding poset $\mathbf{P}$ in a reasonable amount of time, even for large sets of rules.

The following variant of model (CR) does not contain an objective function and is amenable to direct solution via a standard constraint programming solver, such as Gecode:

$$\text{(CP-CR)}\ \ \text{find } z, y \text{ satisfying (4)-(6)},$$
$$z_i \in \{0, 1\},\ i = 1, \ldots, n,$$
$$y_j \in \{0, 1\},\ j = 1, \ldots, |R|.$$

We made use of the data set "Congressional Voting Records", from the UCI Machine Learning Repository (Frank and Asuncion 2011), in order to learn a rule ensemble and carry out the construction of poset $\mathbf{P}$. The data set consists of a binary classification task, with binary observation data and some missing attribute values. For additional information on the data set, we refer the reader to (Frank and Asuncion 2011).

The decision forest was learned from the data set with the use of the Random Forest (Breiman 2001) classifier available in the WEKA software (Hall et al. 2009). A modification was applied to the Java source code of WEKA's *RandomForest* classifier, in order to output a description of the decision trees learned from the data. A post-processed version of this description was then fed to model (CP-CR) and the set of feasible solutions to the model was obtained and recorded.

Figure 2 illustrates one of the decision trees generated for the data set. The classifier achieved 96.8% accuracy in a 10-folding experiment, with a total of 30 trees in the forest, the maximum depth of each tree equal to 5, and the number

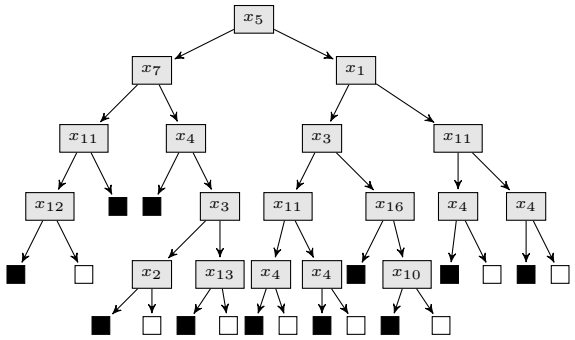

Figure 2: Decision tree in the random forest learned from data set *Voting*. Tests emanating from a node are coded in the following way. The node itself is labeled X_j. Left branches correspond to tests of the type $X_j = 0$, while right branches correspond to $X_j = 1$. As before, black leaves are associated with class $+1$ and white leaves with class -1. This individual tree corresponds to a set of 11 positive rules and 8 negative rules.

of randomly selected features considered in each node being $\lfloor \log_2(n) \rfloor + 1 = 5$, where $n = 16$ is the number of features. The average number of nodes per tree was 47. On the full data set, the accuracy of the classifier was 97.7%. The total number of rules derived from the forest was 720. We remark that, despite the total numer of 720 rules, exactly 30 rules are active at any given observation, given the inherent structure of a decision forest classifier: only one rule from each tree can be active on a given observation.

In order to specify model (CP-CR), an observation $\mathbf{x}$ was selected from the data set. We arbitrarily selected the 411-th observation from the dataset, which contained no missing attribute values and was classified with a large majority of the votes from the trees (all but two trees in the forest classified it as belonging to the "republican" class). In order to confer a high confidence level to the classification reversal, we made use of constraints (9) and (10), restricting the absolute value of v to be at least 20 (i.e., out of the 30 trees, at most 5 should classify the observation as belonging to its original class – i.e.,). Moreover, we limited the number of feature changes to be at most 6. With such parameters, model (CP-CR) admitted a total of 74 solutions, out of which 3 were optimal, with the minimum number of 4 feature changes.

The structure of the resulting poset $\mathbf{P}$ is partially shown in Figure 3. Each element is shown as a rectangle with two numbers: the left-hand side number informs the value of $|\Delta(\mathbf{x}')|$ associated with the solution, while the value on the right is the number of feature changes. The maximal elements of $\mathbf{P}$ had Δ-values ranging from 22 to 26. This means that, in order to get a higher Δ-value (28 or the unanimous 30), it is necesssary to effect more than 6 feature changes. Note that by definition, the structure of $\mathbf{P}$ will be that of a weak (i.e., non-induced) subposet of the Boolean lattice, with some of the lowest and highest elements, as well as some comparability relations, missing.

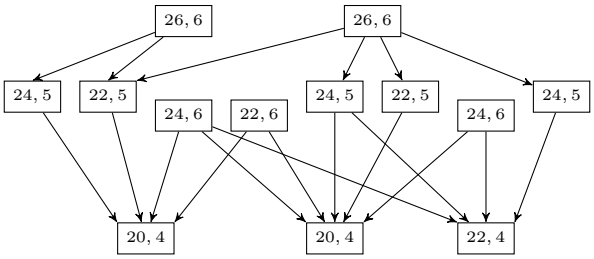

Figure 3: Partial Hasse diagram of poset $\mathbf{P}$ for data set *Congressional Voting*.

8 Conclusions

We introduced the problem of reversing the classification of a binary observation $\mathbf{x}$ by a given weighted rule ensemble F via modifications on the values of features of $\mathbf{x}$. We showed the problem to be $\mathcal{NP}$-complete and proposed a discrete optimization model for the solution of its optimization version.

We also presented the results of preliminary computational tests on a standard classification task from the machine learning literature. If the rule weights are integer, model (CP-CR) can be appropriately formulated using a standard constraint programming engine, allowing the enumeration of the complete set of solutions. The poset of solutions thus generated can be used by a decision maker to guide the selection of an appropriate set of feature modifications to apply to the given observation.

In addition to the potential use in the identification of biomarkers and the consequent development of therapies, the RECR problem described in this paper can be used as part of feature selection/ranking procedures. It can be argued that the relevance of a feature to a set of observations can be estimated on the basis of its relative presence in small-sized solutions to the corresponding set of RECR instances.

As another future line of work we suggest applying model (CP-CR) to a classifier built for a real-life genomic data set. We believe that the following more general version of RECR might be of interest: instead of requiring the reversal of the classification of a single observation $\mathbf{x}$, one can ask for a smallest subset of features which, once fixed at specific values, cause the reversal of the classification of a given set of observations.

9 Acknowledgments

The authors are thankful to Irina Lozina, David Axelrod and Peter L. Hammer for helpful discussions at a preliminary stage of this work. The authors also acknowledge the valuable input of an anonymous referee that helped improve the readability of this paper.

The first author was supported by CNPq, the Brazilian Council for Scientific and Technological Development. The second author was supported by a Scientific Initiation scholarship from the Federal University of the Semi-Arid.

References

Alexe, S., and Hammer, P. L. 2006. Accelerated Algorithm for Pattern Detection in Logical Analysis of Data. *Discrete Applied Mathematics* 154(7):1050–1063.

Axelrod, D.; Bonates, T.; Hammer, P.; and Lozina, I. 2004. From Diagnosis to Therapy via LAD. *Invited Lecture at INFORMS Annual Meeting, Denver, CO*.

Boros, E.; Hammer, P. L.; Ibaraki, T.; and Kogan, A. 1997. Logical Analysis of Numerical Data. *Mathematical Programming* 79:163–190.

Boros, E.; Hammer, P. L.; Ibaraki, T.; Kogan, A.; Mayoraz, E.; and Muchnik, I. 2000. An Implementation of Logical Analysis of Data. *Knowledge and Data Engineering, IEEE Transactions on* 12(2):292–306.

Breiman, L. 2001. Random Forests. *Machine Learning* 45(1):5–32.

Chin, L.; Hahn, W. C.; Getz, G.; and Meyerson, M. 2011. Making Sense of Cancer Genomic Data. *Genes & Development* 25(6):534–555.

Crama, Y., and Hammer, P. L. 2011. *Boolean Functions: Theory, Algorithms, and Applications*, volume 142. Cambridge University Press.

Frank, A., and Asuncion, A. 2011. UCI Machine Learning Repository, 2010. *URL http://archive. ics. uci. edu/ml* 15:22.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The WEKA Data Mining Software: An Update. *SIGKDD Explorations* 11(1).

Ho, T. K. 1998. The Random Subspace Method for Constructing Decision Forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20(8):832–844.

Schapire, R. E.; Freund, Y.; Bartlett, P.; and Lee, W. S. 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics* 26(5):1651–1686.

Schulte, C.; Tack, G.; and Lagerkvist, M. Z. 2010. Modeling and Programming with Gecode.

Statnikov, A., and Aliferis, C. F. 2007. Are random forests better than support vector machines for microarray-based cancer classification? In *AMIA Annual Symposium Proceedings*, 686–690. American Medical Informatics Association.

Trotter, W. T. 2001. *Combinatorics and Partially Ordered Sets: Dimension Theory*. Johns Hopkins University Press.