Proceedings of the 10th International Symposium on Computer and Information Sciences, pp. 411-418, 1995.

NP-Completeness of Minimum Rule Sets

Tim L. Andersen and Tony R. Martinez

Computer Science Department, Brigham Young University, Provo, Utah 84602
e-mail: tim@axon.cs.byu.edu, martinez@cs.byu.edu

ABSTRACT

Rule induction systems seek to generate rule sets which are optimal in the complexity of the rule set. This paper develops a formal proof of the NP-Completeness of the problem of generating the simplest rule set (MIN RS) which accurately predicts examples in the training set for a particular type of generalization algorithm algorithm and complexity measure. The proof is then informally extended to cover a broader spectrum of complexity measures and learning algorithms.

Key Words: Rule Induction, Machine Learning, NP-Completeness.

1 Introduction

This paper discusses the computational complexity of generating minimum complexity rule sets. Several rule induction systems have been proposed in the literature. A few of these include SWAP-1, (Weiss '91; Chidanand '94) AQ inductive algorithms (Michalski '83, '86; Wnek '94), RULES (Pham '95), BBG (Van Horn '93), and CN2 (Clark '90). Other methods for rule induction systems include LAIR (Elio '91), LICHEN (Donoho '94), C4.5 (Quinlan '86), and STAGGER (Schlimmer '86). All of these systems seek to generate rule sets which are optimal in the complexity of the rule set generated by the learning algorithm.

This paper presents a formal proof concerning the NP-Completeness of the problem of generating minimum complexity (simple) rule sets. The proof uses a straightforward transformation from minimum set cover to show that this problem is NP-Complete. This transformation is similar to that used by (Van Horn 1994) for the minimum feature set problem. Due to the nature of this transformation, it can also be shown that the minimum rule set problem is at least as hard to approximate as minimum set cover.

Section 2 defines some of the terms which will be used in the proof. Section 3 discusses the vastness of the hypothesis space which rule induction systems must deal with, and gives the formal proof of NP-Completeness along with extensions. Section 5 gives the conclusion.

2 Terms and Definitions

The following terms and definitions are used throughout the remainder of this paper. First, the idea of the problem domain is defined. Then the definition of an example from the problem domain is given. Next, the definition of a rule is given, and finally, definitions which relate rules to examples are given.

Definition 2.1 D is defined to be the problem domain (the set of all possible data points).
Definition 2.2 O is defined to be the finite set of states into which the individual elements of D may be classified. For $1 \le i \le |O|$, the ith class of O is written as o_i.
Definition 2.3 Each data point $d \in D$ is composed of two parts (in the following a dot notation is used, for example, $d.\boldsymbol{a}$ is written to indicate d's input vector $\boldsymbol{a}$):

a) $d.\boldsymbol{a} = a_1 a_2 \ldots a_m$, an input vector of m nominally valued attributes.

b) $d.\boldsymbol{c} = d.c_1 c_2 \ldots c_{|O|}$, a probability vector of real values on the range [0,1]. The length of the vector is equal to the number of possible output classifications for any pattern in D. Note that while this information is said to exist, it is not necessary that it be known (in general, this information is not known).

Definition 2.4 Given a data point $d \in D$, the individual a_i's of d are commonly called attributes or features. For the purposes of this paper, the number of discrete values which any given attribute a_i of d may take is constant and finite. The finite set of states which a_i may take is written as S_i. Furthermore, each S_i always contains the * state (the asterisk is a special "don't care" state, the use of which is explained in definition 2.10). That is, $* \in S_i$ for all i, $1 \leq i \leq m$.

Definition 2.5 A Data Set E is defined to be a finite set of preclassified examples drawn from a particular problem domain D. Let $e \in E$. Then e is an example from the data set E. For $1 \leq j \leq |E|$ we write e_j to indicate the jth example in E, where the elements of E have been ordered arbitrarily from 1 to $|E|$.

Definition 2.6 Each example $e \in E$ is composed of three parts:

a) $e.\boldsymbol{a} = a_1 a_2 \ldots a_m$, an input vector of m nominally valued attributes.

b) $e.o = o_i$, an output classification which is equal to some $o_i \in O$.

c) $e.id$ = an integer valued identification label. It is required that for all $e, y \in E$ that $e.id = y.id$ if and only if $e = y$. Further, for all $e \in E$, $1 \leq e.id \leq |E|$.

Definition 2.7 A rule f of a data set is composed of three parts:

a) $f.\boldsymbol{a} = a_1 a_2 \ldots a_m$, an input vector of m nominally valued attributes.

b) $f.o = o_i$, a nominally valued output classification with which the rule is associated.

c) $f.p$, a relative strength metric which indicates the degree of correlation that the rule has with the output $f.o$ in relation to other rules.

Definition 2.8 We write F to indicate the current set of all rules for a data set. We define f_j to be the jth rule of F (where the rules in F have been arbitrarily ordered from 1 to $|F|$).

Definition 2.9 A set of rules F is said to be ***defined*** on a problem domain D iff the following is true: $(\forall f)(f \in F \rightarrow (f.o \in O) \wedge (\forall j)((1 \leq i \leq m) \rightarrow f.a_i \in S_i))$.

Definition 2.10 We define a matching function for attributes of rules and examples called *a_match*(a, b) (attribute match) as follows:

define *a_match*(a, b) such that

a_match(a, b) = 1 if $((a = b) \vee (a = *) \vee (b = *))$,

else *a_match*(a, b) = 0.

Definition 2.11 We define a matching predicate for rules and examples r_*match*(f, e) as follows:

r_match(f, e) = True if $\sum_{i=1}^{m} a\_match(f.a_i, e.a_i) = m$,

otherwise *r_match*(F, E) = False.

3 On The Complexity of Learning DNF Rules

The usual approach to learning from examples is as follows: given a data set E containing preclassified examples from a particular problem domain, randomly partition the data set into two parts, a training set T and a test set V. Run the learning algorithm in adaptive mode using only the examples in the training set T. Next, run the learning algorithm in nonadaptive mode on the test set V to determine generalization accuracy.

During the adaptive phase, the learning algorithm attempts to extract critical rules from the examples in the training set. In the search for potentially interesting rules a realistic learning algorithm will have to ignore certain, potentially vast areas of the search space, in particular with problems that have large numbers of input attributes. The first question we may ask is can we guarantee that ignoring certain areas of the search space will not adversely affect the solution generated by our learning algorithm? If the answer to this questions is no, then another question we may wish to ask is that if areas of the search space are ignored, how harmful will this be to the solution? This paper attempts to address these issues in this section.

3.1 The Vastness of the Hypothesis Space

We begin with a discussion of the actual size of the hypothesis space. The functional mapping of input vectors to output classification derived by the learning algorithm is called the hypothesis. Different learning algorithms may have vastly different ways of representing hypotheses. For example, many rule induction algorithms represent hypotheses with sets of rules that are in disjunctive normal form, while neural networks represent hypotheses with networks of nodes that are connected via weighted links. In general, the set of all possible hypotheses on a particular problem domain which are representable by a given learning algorithm is extremely large, so large, that it simply is not feasible to check every possible hypothesis in order to determine which is the best.

For example, let us consider the DNF rule set method for representing hypotheses. Let R be the set of all possible rules, then the size of the set of all possible rules is exponential in the number of input attributes, and can be calculated from the following formula:

$$|R| = \prod_{i=1}^{m} |S_i|.$$

Furthermore, each and every rule in the set of all possible rules is available for selection by the learning algorithm as a member of the final set of rules which will be used for decision making. The number of different possible rule sets which the learning algorithm can generate is thus the power set of the set of all possible rules. The size of this set is $2^{|R|}$. Notice that this is already a very large number (super-exponential). Let us assume that the learning algorithm organizes the selected rules into an ordered list. For any given set of rules F, the number of different ways to order the rules is $|F|!$. Thus, the number of different hypotheses theoretically expressible by the learning algorithm is:

$$\sum_{i=1}^{2^{|R|}} |F_i|!,$$

where F_i represents the ith possible rule set, the set of all possible rule sets having been ordered from 1 to $2^{|R|}$. Thus, with even a relatively small number of input attributes the number of different hypotheses theoretically expressible by the learning algorithm can quickly outpace the number of atoms in the galaxy.

Simply demonstrating the size of the hypothesis space for this approach does not prove how difficult it might be to find the "best" hypothesis. Indeed, it might be hoped that in spite of the extremely large hypothesis space, there might exist a polynomial time algorithm which can deterministically find the optimum hypothesis.

3.2 Minimum Rule Set

In general, when given a choice between two or more hypotheses which exhibit the same degree of classification accuracy, the most preferable hypotheses is the one which is the simplest. Let F_1, F_2 be sets of rules, then for the purposes of this paper we define F_1 to be simpler than F_2 if $|F_1| < |F_2|$. The weakness of this definition is that F_1 might not really be simpler than F_2 in the case where $F_1 \not\subset F_2$, even though $|F_1| < |F_2|$, since the simplicity measure is not taking into account the complexity of the individual rules. However, the strength of this measure of simplicity is that it is a relatively straightforward and intuitive notion that is easily measured. For the proof of NP-completeness in this paper we consider F_1 to be simpler than F_2 if $|F_1| < |F_2|$, even if $F_1 \not\subset F_2$. We then informally extend the proof to the problem of determining the simplest set of rules when individual rules have an associated complexity, which can also be seen to be NP-Complete in the general case.

More formally, suppose that we have a set of preclassified examples T from a problem domain D, and that we have a set of rules F which we may use to classify these examples.

Suppose further that we have a generalization algorithm *gen* which takes as input examples from T and a set of rules F, and classifies elements of T based upon the rules in F. Given an example e, the output of the function $gen(e,F)$ is said to be defined if there is an $f \in F$ such that f matches e, otherwise the output of $gen(e,F)$ is undefined. Other than this requirement, there need not be any other restrictions on how *gen* may use the rules in F.

Given this information we ask the question, is there some subset $F' \subset F$ such that the output of $gen(e,F')$ is correct (the error of classification is minimized) for all $e \in T$? If so, what is the smallest such subset of F? Alternately, one might ask the (weaker) question, is there some subset $F' \subset F$ such that the output of $gen(e,F')$ is *defined* for all $e \in T$ (in this case we don't care if the predicted classification is correct, only that some classification is predicted)? And if so, what is the smallest such subset of F? It is shown in this paper that both of these questions are NP-complete.

3.2.1 Generalization Algorithm

For the purposes of the proof given in this paper, it is not necessary that any specific generalization method for resolving conflicts between matched rules be defined. It is only required (for the proof) that for each example there exists at least one rule which matches the example and correctly predicts that example's classification.

Although no specific generalization method is required, for the purposes of illustration we define a generalization scheme in figure 5.1 which uses a voting method that sums the metrics of all matched rules for the respective possible output classifications, then chooses the output classification with the greatest sum.

> Given a set E of examples and a set F' of rules, for each $e \in E$, choose the predicted output classification o_i which maximizes the summation
>
> $$\sum_{f \in F' \wedge f\_match(f,e) \wedge f.o = o_i} f.p$$
>
> If there exists an $e \in E$ such that there is no $f \in F'$ which matches e, then the output of the generalization algorithm is said to be undefined.

Figure 5.1. Generalization algorithm

The proof is then defined for any learning algorithm which satisfies the following. Given a set T of preclassified training examples and a set F of rules derived from T, return a set of rules $F' \subseteq F$ such that $(\forall e)(e \in E \rightarrow gen(e,F') = e.o)$. Alternately, in the case where there are inconsistencies in the set of examples, we would choose the set of rules F' which minimizes the error of misclassification.

This implies that the learning algorithm returns a set of rules F' such that the output of the generalization algorithm is always defined for each example $e \in T$. Further, for each example $e \in T$, there will be a rule $f \in F'$ which matches e and correctly predicts e's output classification. In other words, the following is true of the set of rules F' returned by the learning algorithm: $(\forall e)(\exists f)(e \in T \rightarrow f \in F' \wedge r\_match(f,e) \wedge e.o = f.o)$.

3.2.2 Proof of NP-Completeness

We first begin by defining the problem of minimum set cover, called MIN SC, a known NP-Complete problem.

MINIMUM SET COVER (MIN SC)
INSTANCE: = (X, C, k), a collection C of subsets of a set X, and a positive integer k.
THE PROBLEM: find some $C' \subseteq C$ such that $|C'| \leq k$ and

$$\bigcup_{C_i \in C'} C_i = X.$$

This problem is known to be NP-complete.

We now formally define the problem of minimum rule set, called MIN RS.

MINIMUM RULE SET (MIN RS)
INSTANCE: = (T,F,k), a set of training examples T, a set of rules F, and a positive integer k.
THE PROBLEM: Find some $F' \subseteq F$ such that $|F'| \leq$ k and $(\forall e)(e \in T \rightarrow gen(e, F') = e.o)$.
Note that, as stated above, this means the following condition (as a minimum requirement) must hold: $(\forall e)(\exists f)(e \in T \rightarrow f \in F' \wedge r\_match(f, e) \wedge e.o = f.o)$.

Lemma 1 *MIN RS* $\in$ *NP*:

Proof. Given an instance (T,F,k) of MIN RS, a non-deterministic algorithm need only guess a subset F' of F of size $\leq k$, and then run the generalization algorithm on each example in T to determine if the condition $(\forall e)(e \in T \rightarrow gen(e, F') = e.o)$ holds true. Assuming that the function *gen* is bounded by a polynomial, then the time bound for this process will also be a polynomial. Thus MIN RS $\in$ NP.

Lemma 2 *There is a function* μ *which maps instances of MIN SC to instances of MIN RS which has the following properties:*

1. *The function* μ *is computable in polynomial time.*
2. *If there is cover* $C' \subseteq C$ *for* X *such that* $|C'| \leq k$, *then there is a rule set* $F' \subseteq F$ *such that* $|F'| \leq k+1$ *and* $(\forall e)(e \in T \rightarrow gen(e, F') = e.o)$
3. *If there is a rule set* $F' \subseteq F$ *such that* $|F'| \leq k+1$ *and* $(\forall e)(e \in T \rightarrow gen(e, F') = e.o)$, *then there is a cover* $C' \subseteq C$ *for* X *such that* $|C'| \leq k$.

Proof. (In the following proof we assume that C contains a cover for X). Given an instance (X,C,k) of MIN SC we define the following mapping function μ to an instance (T,F,k) of MIN RS:

In the following transformation, the elements of X correspond to the examples in T, and the elements of C correspond to the rules in F. Let $n = |X|+1$, $m = |C|+1$. Enumerate the elements of X and C. Then $X = \{x_1, x_2, \ldots, x_{n-1}\}$, and $C = \{C_1, C_2, \ldots, C_{m-1}\}$.

For each element in X, we create example one example in T. Let $T = \{e_1, e_2, \ldots, e_n\}$, where for $1 \leq j < n$, e_j in T corresponds to element x_j in X. For each $e_j \in T$, where $1 \leq j < n$, let

$e_j.\boldsymbol{a} = a_1 a_2 \ldots a_m$, where
$a_d = 1$ if $x_j \in C_d$
else $a_d = 0$.
$e_j.o = 1$.

In addition, we create an example in T which does not correspond to any element of X.
$e_n.\boldsymbol{a} = a_1 a_2 \ldots a_m$, where $a_d = 0$ for all a_d in $e_n.\boldsymbol{a}$
$e_n.o = 0$.

Let $F = \{f_1, f_2, \ldots, f_m\}$, where for $1 \leq j < m$, f_j in F corresponds to set C_j in C (in other words, there is exactly one f_j in F for each C_j in C). For each $f_j \in F$, where $1 \leq j < m$, let

$f_j.\boldsymbol{a} = a_1 a_2 \ldots a_m$, where
$a_d = 1$ if $j=d$
else $a_d = *$.
$f_j.o = 1$

In addition, we create a catch-all rule which is guaranteed to match any example.

$f_m.\boldsymbol{a} = a_1 a_2 \ldots a_m$, where $a_d = *$ for all a_d in $f_m.\boldsymbol{a}$
$f_m.o = 0$.

We first show that the mapping function fulfills the first condition. The time required to create the set of training examples T is bounded above by $O(mn^2)$. There are n examples in T. Each example has a vector of length m. For each element of the vector, the time required to determine if $x_j \in C_k$ is at most n-1, where n-1 is the maximum size of any subset of X (since $n = |X|+1$). Thus an upper bound for the creation of T is $O(mn^2)$.

The upper bound for the creation of F is $O(m^2)$. There are m rules in F. Each rule in F has a vector of length m. The operation $j=k$ takes time $O(1)$. Therefore we have the upper bound of $O(m^2)$ to create F. From above we have $O(m^2 + mn^2)$ as the upper time-bound for the mapping function. This upper bound thus fulfills the first condition that the mapping can be computed in polynomial time.

We now show that the mapping fulfills the second requirement. Assume that there is a cover C' such that $C' \subseteq C$ and $|C'| \leq k$. Select $F' \subseteq F$ where $F' = \{f_j \mid C_j \in C'\} \cup \{f_m\}$, then $|F'| \leq k+1$. For all f_j, where $j < m$, let $f_j.p = 2$. Let $f_m.p = 1$.

From the construction we know that the only rule which matches example e_n is rule f_m (in other words, $\{f \mid f \in F \wedge f\_match(f_j, e_n)\} = \{f_m\}$). Since $f_m.o = 0$ and $f_m.p = 1$ we have

$$\sum_{f \in F' \wedge r\_match(f, e_n) \wedge f.o = 0} f.p = 1$$

$$\sum_{f \in F' \wedge r\_match(f, e_n) \wedge f.o = 1} f.p = 0$$

which implies that $gen(e_n, F') = e_n.o = 0$.

We also know from the construction that for all $1 \leq j < m$, $\{x_i \mid e_i \in T \wedge r\_match(f_j, e_i)\} = C_j$. Therefore, since C' is a cover for X, and for each $C_j \in C'$ we have $f_j \in F'$, we know that

$$\bigcup_{f_j \in F', j \neq m} \{x_i \mid e_i \in T \wedge r\_match(f_j, e_i)\} = X.$$

Therefore, $$\bigcup_{F_j \in F', j \neq m} \{e_i \mid e_i \in T \wedge r\_match(f_j, e_i)\} = T - e_n.$$

This means that for each example $e \in T$ (with the exception of example e_n) there is at least one rule $f \in F'$ with $f.o = 1$ and $f.p = 2$ which matches e. In addition, we know that rule f_m matches all $e \in T$. Therefore the following equations hold true for all $e \in T$, where $e \neq e_n$.

$$\sum_{f \in F' \wedge r\_match(f, e) \wedge f.o = 0} f.p = 1$$

$$\sum_{f \in F' \wedge r\_match(f, e) \wedge f.o = 1} f.p \geq 2$$

which implies that for all $e \in T$ where $e \neq e_n$, $gen(e, F') = 1$, thus fulfilling the second property.

Lastly, we show that the mapping has the property that if there is a rule set $F' \subseteq F$ such that $|F'| \leq k+1$ and $(\forall e)(e \in T \rightarrow gen(e, F') = e.o)$, then there is a cover $C' \subseteq C$ such that $|C'| \leq k$. Assume that there is a set of rules $F' \subseteq F$ such that $|F'| \leq k+1$ and $(\forall e)(e \in T \rightarrow gen(e, F') = e.o)$. Since rule f_m is the only rule which matches example e_n, this rule must be assigned a positive metric, and be included in the set F' in order for example e_n to be classified correctly. Let $f_m \in F'$ and $f_m.p = x$, where x is a positive real number. Since we have assumed that all examples from T are classified correctly, we know that for all $e \in T$, where $e \neq e_n$, there must be some $f_j \in F'$ which matches e, and such that $f_j.o = 1$ and $f_j.p > x$. We also know that in this case $j \neq m$ (since $f_m.o = 0$).

This implies that for each such f_j there is a corresponding $C_j \in C$ (since rule f_j in F, $j < m$, corresponds to set C_j in C from the construction). Let $C' = \{C_j \mid f_j \in F' \wedge j \neq m\}$. From the above paragraph we know that $(\forall i)(\exists j)(e_i \in T \wedge i \neq n \rightarrow f_j \in F' \wedge j \neq m \wedge r\_match(f_j, e_i))$. Then this implies that $\{e_i \mid e_i \in T \wedge f_j \in F' \wedge j \neq m \wedge r\_match(f_j, e_i)\} = T - \{e_n\}$, and thus $\{x_i \mid e_i \in E \wedge f_j \in F' \wedge j \neq m \wedge r\_match(f_j, e_i)\} = X$.

Furthermore, from the construction we have that for any j such that $1 \geq j \geq m$, $C_j = \{x_i \mid e_i \in T \wedge r\_match(f_j, e_i)\}$. Therefore,

$$\bigcup_{C_j \in C'} C_j = \left\{x_i \mid e_i \in T \wedge f_j \in F' \wedge j \neq m \wedge r\_match(f_j, e_i)\right\} = X.$$

Therefore C' is a cover for X of size $\leq k$.

Theorem 1 MIN RS is NP-complete.

Proof. By Lemma 1, we know that MIN RS is in NP. By lemma 2, we know that there is a mapping function which maps instances from a known NP-complete problem (MIN SC) to instances of MIN RS which can be computed in polynomial time. Therefore, MIN RS is NP-complete.

3.2.3 Discussion on the NP-completeness of MIN RS

The proof above can also be used to show that the question of determining the smallest set of rules $F' \subseteq F$ such that $(\forall e)(\exists f)(e \in T \rightarrow f \in F' \wedge r\_match(f, e) \wedge e.o = f.o)$ is NP-complete. The motivation for this question is that we assume that *gen* does not have a chance of correctly predicting the classification for an example e if there is not at least one rule in F' with the appropriate output which matches e, and so we are asking, "can we find the smallest set of rules which has even a remote chance of correctly classifying examples from T?" Intuitively, this question would seem easier to answer than that of MIN RS, since we no longer have the requirement to check the output of the generalization algorithm. However, this question is also NP-complete.

With some modification, the proof can also be extended to show that the even weaker question of determining the smallest set of rules $F' \subseteq F$ such that $(\forall e)(\exists f)(e \in T \rightarrow f \in F' \wedge r\_match(f, e))$ is NP-complete. We know that the output of the generalization algorithm is undefined for an example e if it does not have any rules which match e, so as a minimum requirement we know that for each example e in T we must have at least one rule in F' which matches e. Otherwise the output of *gen* is guaranteed to be incorrect since it will be undefined. Thus, the NP-completeness of this question implies the NP-completeness of MIN RS, since MIN RS contains this problem as a sub-problem (in other words, there are problems where we will have to be able to answer this question in order to be able to answer MIN RS).

The actual output of the generalization algorithm *gen* is not referenced in either of the two NP-complete questions stated above. This implies that no matter how the generalization algorithm utilizes rules to predict classification, the general question of MIN RS is NP-complete regardless of the exact form of the generalization algorithm. This holds as long as we require that $(\forall e)(\exists f)(e \in T \rightarrow f \in F' \wedge r\_match(f, e))$ in order for the output of *gen* to have a chance of being correct.

The NP-completeness of MIN RS is not dependent upon the format used for rules or examples given in this paper. For any rule format, the set of examples that a given rule matches is always a subset of the total set of examples. We can define this set to be $\{e \mid e \in T \wedge r\_match(f, e)\} \subseteq T$ for any given rule f, where *r_match* is a deterministic matching function. Thus, for this paper, we could have defined the following mapping from instances of MIN SC to MIN RS for the rules in MIN RS:

For each $f_j \in F$, where $1 \leq j < m$, let

$r\_match(f_j, e_i) = 1$ if $x_i \in C_j$
else $r\_match(f_j, e_i) = 0$

$f_j.O = 1$

For $j = m$ let

$r\_match(f_j, e_i) = 1$ for all $e_i \in T$

$f_j.O = 0$.

Note that with this mapping the set of examples which each rule matches is the same as in the mapping given in the proof to theorem 1. However, the mapping does not depend upon any specific formulation for the rules or for the examples, nor does it specify the exact inner workings of the function *f_match*. Thus, the results of this paper can be generalized to any rule-based system where examples are classified by first matching them with rules via some type of deterministic matching function, and then using those rules to classify the examples.

It can also be seen that the problem of determining the simplest set of rules which accurately describes a given data set is (in general) still NP-Complete, even when the complexity of individual rules is included in the calculation of the over-all complexity of a given set of rules. For example, in the proof of theorem 1 the rules f_1 through f_{m-1} all have exactly one input value specified, with all other input variables set to a "don't care" state, while rule f_m has all of its variables set to the "don't care" state. Thus, a reasonable rule complexity measure might assign each of the rules f_1 through f_{m-1} the same complexity value (since each of these rules requires a test on exactly one of the input variables - all input variables being 3-state variables), and would assign a lower complexity value to rule f_m (since it does not require a test on any variables to determine a match - it always matches any example). In this case the proof of theorem 1 remains unchanged, since the set F' will remain unchanged. We have already shown that the set F' must contain rule f_m, since it is the only rule which covers example e_n. The remaining rules are then of equal complexity, and we are thus free to choose rules from the remaining set of rules without the need to worry about the complexity of individual rules. The set of rules F' which we obtain in this case is then equal to the set of rules F' which we obtain in theorem 1 (in other words, if F' is an answer for MIN RS, then F' will also be an answer in this case, and visa versa).

4 NP-Completeness Conclusion

The problem of determining the smallest set of rules which covers a set of examples and correctly predicts their classification has been shown to be NP-Complete for a specific type of rule and generalization algorithm. It has also been shown how the more general problem, where the specific form of the rules and generalization algorithm is not exactly defined, is NP-Complete given a few reasonable assumptions about how rules match examples, and how the generalization algorithm uses those rules to classify examples. The assumptions are that the matching of rules with examples must be deterministic, and the output of the generalization algorithm is undefined in the case where an example has no matching rules.

Thus, it is necessary to use an algorithm which returns an approximate solution for this problem, since an algorithm which returns the optimal solution will take time exponential in the size of the problem. Due to the nature of this transformation from MIN SC to MIN RS, it can also be shown that in addition to being NP-Complete, the problem of approximating MIN RS is at least as difficult as the problem of approximating MIN SC. The current best approximating algorithm for MIN SC is only guaranteed to approximate MIN SC to within a factor $ln|X| + 1$ of the size of the optimal solution (not a constant factor). This implies that it could be very difficult to find any approximating algorithm for MIN RS which returns a result better than a log factor of optimal.

5 Bibliography

Chidanand, A., F. Damerau, and M. W. Sholom. Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems,* Vol. 12, No. 3, July 1994, pp 233-251.

Clark, P., and T. Niblett. The CN2 Induction Algorithm. *Machine Learning,* Vol. 45, 1990, pp. 304-307.

Donoho, Steven K. and David C. Wilkins. Using Apprenticeship Techniques to Guide Constructive Induction. *Knowledge Acquisition*, vol 6, pp 295-314, 1994.

Elio, Renee and Larry Watanabe. An Incremental Deductive Strategy for Controlling Constructive Induction in Learning from Examples. *Machine Learning*, vol 7, pp 7-44, 1991.

Michalski, R. S., I. Mozetic, J. Hong, and N. Lavrac. The Multi-purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. *Proceedings of the Fifth National Conference on Artificial Intelligence*, pp. 1041-1045, Philadelphia: Morgan Kaufmann, 1986.

Michalski, R. S., and J. Larson. Incremental Generation of VL_1 Hypotheses: The Underlying Methodology and the Description of the Program AQ11. *Technical Report ISG 83-5*, Urbana: University of Illinois, Computer Science Department, 1983.

Pham, D. and M. Aksoy. RULES: A Simple Rule Extraction System. *Expert Systems with Applications*, Vol. 8, No. 1, pp. 59-65, 1995.

Schlimmer, Jeffrey C. and Richard H. Granger. Incremental Learning from Noisy Data. *Machine Learning*, vol 1, pp 317-354, 1986.

Van Horn, Kevin S and Tony R. Martinez. The BBG Rule Induction Algorithm. *Proceedings of the 6th Autstralian Joint Conference on Artificial Intelligence*. World Scientific Publishing Co., Singapore 1993.

Van Horn, Kevin S and Tony R. Martinez. The Minimum Feature Set Problem. *Neural Networks*. vol. 7, no. 3, pp. 491-494, 1994.

Weiss, Sholom and N. Indurkhya. Optimized Rule Induction. *IEEE Expert,* December 1993, 61-69.

Wnek, Janusz and Ryszard S. Michalski. Hypothesis-Driven Constructive Induction in AQ17-HCI: A Method and Experiments. *Machine Learning*, vol 14, pp 139-168, 1994.