Mining Predictive k -CNF Expressions

Anton Dries, Luc De Raedt, Siegfried Nijssen

Abstract—We adapt Mitchell’s version space algorithm for mining k -CNF formulae. Advantages of this algorithm are that it runs in a single pass over the data, is conceptually simple, can be used for missing value prediction, and has interesting theoretical properties, while an empirical evaluation on classification tasks yields competitive predictive results.

Index Terms—Concept learning, Machine learning, Data mining.

1 INTRODUCTION

ONE of the most popular data mining tasks is rule-based classification. There are many popular rule-based classification methods. Some of these learn rule sets directly, for instance, RIPPER [1], PART [2] and CPAR [3], whereas other ones post-process a decision tree [4], or construct sets of rules by post-processing association rules, for instance, CBA [5] and CMAR [6]. All these algorithms use different strategies to find and use sets of rules for classification. A common feature is, however, that they learn classifiers in *disjunctive normal form* (DNF), that is, a prediction of the classifier is based on the truth of a formula of the following form:

$$(a_{i_{11}} \wedge a_{i_{12}} \wedge \dots a_{i_{1k}}) \vee (a_{i_{21}} \wedge a_{i_{22}} \wedge \dots a_{i_{2k}}) \vee \dots$$

where every conjunction corresponds to a rule and it is sufficient if one rule matches. For instance, in a decision tree, each path to a leaf can be transformed into a conjunctive formula. All leaves predicting the same class label can be considered part of one disjunction for that class. The resulting DNF formula is true for a certain class iff the tree predicts that class. In other approaches this idea is adapted by attaching weights to rules, or ordering them, but rules are always conjunctive.

In this paper, we revisit the algorithm of Valiant for learning boolean functions [7] and employ it to learn a set of clauses. This algorithm, which can be seen as a variant of Mitchell’s Candidate Elimination algorithm [8], is typically only used for theoretical purposes. In contrast to most rule-learners, it can be used to derive a formula in *conjunctive normal form* (CNF), that is, a formula of the following kind:

$$(a_{i_{11}} \vee a_{i_{12}} \vee \dots a_{i_{1k}}) \wedge (a_{i_{21}} \vee a_{i_{22}} \vee \dots a_{i_{2k}}) \wedge \dots$$

Every conjunct in a CNF formula is usually called a *clause*. A well-known theoretical result is that a k -CNF formula, that is, a CNF formula in which each clause has at most k literals, is *PAC-learnable*: with high probability a hypothesis can be learned that has low error rate using a polynomial number of training examples. In contrast with DNF-based learning algorithms, the CNF algorithm follows a bottom-up strategy. It starts from the most specific set of rules, i.e., one that covers no examples, and gradually generalizes this set to cover each (positive) example in the dataset. This

approach makes it very well suited for learning descriptive rules from positive examples only, which is proven to be impossible with DNF-based learners [7]. CNF-based concepts have also been shown to be competitive with DNF-based concepts in classification tasks [9].

In addition to these theoretical advantages, the CNF learning algorithm also has several important practical advantages:

- it is incremental: one pass over the data is sufficient to build the classifier. Therefore, the algorithm can easily be applied on large datasets common in data mining applications and on streaming data.
- it is not needed to fix the class attribute and it can be used for missing value prediction.
- it is simple: the algorithm has only one parameter (the size of the clauses learned), and rules are interpretable.

Despite these interesting properties, the CNF learning approach has only rarely been used in practice (but see [10], [11]). In this paper, we demonstrate that the technique is also useful in practice and competitive with other rule learning algorithms, hence deserving further consideration.

As compared to other data mining approaches, further distinguishing features of our k -CNF algorithm are (1) that disjunctions contain both positive and negative literals, and (2) that only disjunctions consistent with *all* examples are considered, eliminating the need for the support threshold that is found in pattern mining approaches such as CMAR and CBA. Similar to CBA and CMAR, the k -CNF learning algorithm learns a set of rules through complete search. Because it is not feasible to list all disjunctions even if a size constraint is employed, we use *boundary sets*, which are common in both machine learning (where they have often been studied in conjunction with the concept of version spaces [8]) and data mining (where they have been used for computing, for instance, maximal frequent itemsets [12]). We study how to compute boundary sets and how to use them for performing classification. Our main contributions are that (1) we consider formulas in CNF instead of DNF; (2) we propose new schemes for weighing rules and performing predictions, and (3) we perform an extensive experimental evaluation of these schemes to validate their use.

The paper is organized as follows. In Section 2 we introduce the problem of mining clauses. In Section 3 we

review Mitchell’s algorithm for determining the border of a version space, and introduce our optimizations to this algorithm. In Section 4 we study strategies for using clauses in classification tasks. In Section 5 we report on experiments and in Section 6 we conclude.

2 PROBLEM SETTING

Let us first introduce the problem of mining clauses. We assume that we are given a database having n examples (or transactions) $T_1, \dots, T_n$, each of which contains a subset of items from a set $\mathcal{I}$, $T_i \subseteq \mathcal{I}$.

We are searching for a classifier that can be represented as a CNF formula. A clause in a CNF formula contains a set of positive and negative literals formed from items taken from $\mathcal{I}$, for instance, the set $X = \{a, \neg b, c, \neg d\}$. This set of literals is interpreted as a disjunctive formula, in our example,

$$a \vee \neg b \vee c \vee \neg d \text{ with } a, b, c, d \in \mathcal{I}.$$

We denote the disjunctive interpretation of a set of literals X by $\vee X$. A k -clause is a clause with at most k literals.

A clause $\vee X$ *covers* (or is *satisfied* by) a transaction, denoted by $T_i \models \vee X$, iff $T_i \cap X \neq \emptyset$, where

$$T_i^* = T_i \cup \{\neg a \mid a \in \mathcal{I}, a \notin T_i\},$$

which corresponds to the set of literals containing an additional negative literal for every item not included in the original transaction.

Clauses differ from the more well-known *itemsets* [13] in two ways. First, clauses also contain *negated* literals; second, clauses are interpreted as disjunctions and not as conjunctions. The conjunctive interpretation of a set of literals can be written as $\wedge X$; for itemsets, $T_i \models \wedge X$ iff $X \subseteq T_i^*$.

We wish to learn a theory in k -CNF form. Hence, we wish to learn a set of k -clauses whose conjunction is true for *all* transactions. This implies that every clause needs to be true on *every* example. We call such k -clauses *true clauses*, denoted by $\mathcal{C}_n^*$. More formally:

$$\mathcal{C}_n^* = \{X \mid \vee X \text{ is a } k\text{-clause} \wedge \forall 1 \leq i \leq n : T_i \models \vee X\}.$$

Compared to frequent itemset mining [13], there is no support threshold, or the support is 100%. The number of clauses in this set can be very high, $3^{\#\mathcal{I}}$ in the worst case.

It is useful to limit the number of clauses by computing only the shortest true clauses, that is, those clauses that do not have any true subclauses. We call such rules *minimal* rules by analogy with maximal itemsets in itemset mining. The set of all minimal true clauses $\mathcal{S}$ constitutes the boundary of $\mathcal{C}^*$. This can be justified as follows. The space of clauses is structured by a generality relation: a clause $\vee X'$ is *more general than* a clause $\vee X$ iff $X \subseteq X'$. Under this generality relation, the coverage relation is *monotonic*:

Property 1 (Monotonicity of clauses): If clause $\vee X'$ is more general than clause $\vee X$, and $T_i \models \vee X$, then $T_i \models \vee X'$.

Intuitively, this means that whenever a clause is true, all its supersets are also true. Hence, there must be a boundary

between true and failing clauses. The (positive) boundary (or S-set) $\mathcal{S}_n$ is defined as follows.

$$\mathcal{S}_n = \{X \mid X \in \mathcal{C}_n^*, \forall \text{ specializations } X' \text{ of } X : X' \notin \mathcal{C}_n^*\}$$

In subsequent sections we study how to compute $\mathcal{S}_n$ incrementally, and how to use it for classification.

3 FINDING CLAUSES

Our algorithm for mining k -CNF formulae is based on the candidate elimination algorithm of Mitchell [8]. The algorithm maintains the S-set $\mathcal{S}_n$ incrementally while reading transactions. Given a transaction T_n , it proceeds as follows:

- 1) initially, $\mathcal{S}_n = \mathcal{S}_{n-1}$;
- 2) compute all clauses in $\mathcal{S}_n$ that are not satisfied by T_n and remove them from $\mathcal{S}_n$;
- 3) compute all minimally general generalizations of these clauses that are satisfied by T_n ;
- 4) verify that these new clauses are minimal, and only if so, add them to $\mathcal{S}_n$.

An important difference with Mitchell’s algorithm is that we only maintain one boundary set, the S-Set. We now discuss in more detail how we implement the different steps of this abstract algorithm efficiently in our system.

3.1 Finding Clauses that Fail

By applying set theory (if $X, Y \subseteq Z$, then $X \cap Y = \emptyset \Leftrightarrow X \subseteq Z - Y$) on the definition of $T_n \models \vee X$, we can derive that

$$T_n \not\models \vee X \text{ iff } X \subseteq \overline{T_n^*},$$

where $\overline{T_n^*}$ denotes all literals over $\mathcal{I}$ not included in T_n^* . Hence, we can find failing clauses for a new example T_n by searching all sets in $\mathcal{S}_n$ that are contained in $\overline{T_n^*}$. Similar to most implementations of the itemset miner Apriori [13], we can store the clauses in a trie, and search the trie for all subsets of $\overline{T_n^*}$.

3.2 Generalizing Clauses

A *minimally general generalization* of a clause $\vee X$ is a clause $\vee X'$ such that $X \subset X'$ and there is no clause Y for which $X \subset Y \subset X'$. Given a clause that is true for all examples T_i with $i < n$, but that is not satisfied by T_n , all least general generalizations that are satisfied by T_n can be computed using $\rho(X)$,

$$\rho(X) = \{X \cup \{x\} \mid x \in T_n^*, \neg x \notin X, x \notin X\}$$

as the following properties hold:

Every $X' \in \rho(X)$ is satisfied by all examples T_i seen earlier ($i < n$). This follows from the monotonicity of clauses.

Every $X' \in \rho(X)$ is satisfied by T_n . This follows from the fact that $T_n^ \cap X' \neq \emptyset$.*

The set $\rho(X)$ contains all minimally general generalizations of X in $\mathcal{C}_n^$. Clause $\vee X$ is a subset of $\overline{T_n^*}$ (because it is not satisfied by T_n). So, if we add a literal $x \notin T_n^*$ then $X \cup \{x\}$ is still a subset of $\overline{T_n^*}$ and therefore will not be satisfied by T_n .*

To restrict the search to consider only k -clauses, the generalizations that contain too many literals are ignored.

3.3 Determining the Boundary Set

A clause $X' \in \rho(X)$ should not be part of the S-Set if X' is the generalization of another clause. After the failing clauses have been removed, we therefore only insert true generalizations for which no subset exists in the current ruleset. To find this redundancy we use the same subset search as in the first step.

3.4 Optimizations

We introduced two further optimizations in the algorithm.

a) Reducing the Number of Literals: This optimization aims at limiting the number of literals to only those that are present in the examples seen so far. When no examples seen so far contain the literal x , then every example satisfies the clause $\neg x$. By keeping track of all the literals that did appear already, we can insert the clause $\neg x$ every time we encounter a new literal x . This also implies that we do not need to know the set of literals $\mathcal{I}$ in advance.

b) Non-Binary Data: Many datasets contain non-binary categorical attributes. We represent such attributes by mapping every value to a separate binary attribute, that is, we insert item a_i in a transaction iff the value of the original, non-binary attribute a is i . However, clauses containing expressions of the kind $\neg a_i \vee \neg a_j$ will always be true and are therefore redundant. We take this knowledge into account and filter such clauses away.

4 CLASSIFICATION FROM CLAUSES

We investigate two ways to use the above approach for classification problems.

Multistream. Given a dataset D consisting of examples belonging to m possible classes, we split the dataset into m datasets D_i (containing only examples belonging to class i) and learn an S-Set for each of the datasets D_i . When a new transaction arrives, we match it with the different S-Sets and classify the example in the class corresponding to the S-Set that best matches the new transaction. This approach corresponds to working with m different streams, one for each of the classes.

Class Completion. Here we assume that the single input dataset contains a special attribute and the task is to predict the value of this attribute. For this setting, we first compute one S-Set for the whole dataset, which includes the class attribute. When we encounter a transaction with a missing class value, we fill in all possible values for this class attribute, and determine how well the clauses in the S-set are satisfied by each completed example. The best matching value is chosen as the new class label. Note that in this setting we do not need to know which attribute is the class attribute while mining.

Both settings employ a metric to determine how well an S-Set matches a transaction. The simplest metric we can use counts the absolute or relative number of clauses that fail for the given transaction (*count metric*).

Unfortunately, there need not be a direct relationship between the number of failing clauses in a boundary representation and the total number of clauses that fail: if a small

clause fails, this implies that many other clauses containing this clause will also fail, and it is not feasible to compute the total number of clauses that fail.

We address this issue by taking the length of the failing rules into account. The formula of this *length* metric is

$$\sum_{i=1}^k \frac{1}{i} \frac{failing_i}{total_i},$$

where $failing_i$ is the number of clauses of length i that fail on the example and $total_i$ the total number of clauses of that length in the results.

5 EXPERIMENTAL EVALUATION

In our experiments we address the following questions:

- 1) How sensitive is the method to choosing the value for the parameter k ?
- 2) How does the predictive accuracy of our algorithm compare to state-of-the-art classification algorithms?
- 3) Can we use our algorithm to complete missing values accurately?

5.1 Question 1

Our algorithm has one parameter, k , which determines the maximal length of the clauses to be mined. Setting this parameter too high will cause the generation of a very large number of clauses and thus high memory usage and processing time. Figure 1(a) shows the effect of k on the runtime of the algorithm for a single dataset averaged over 10 runs of training (on 90% of the dataset). Figure 1(b) shows the effect of k on the predictive accuracy for the same dataset, using the length metric in the class completion setting.

Fig. 1. Effect of k on runtime and accuracy for the chess kr-vs-kp dataset (runtimes for Pentium D, 2.8 GHz, 2Gb RAM)

For this dataset we see that there is a large increase in runtime for $k > 5$. The predictive accuracy for the same dataset however does not improve for higher values of k . We repeated this experiment for four more datasets, shown in Figure 2, and got similar results. We therefore set $k = 5$ in all further experiments, in which we used 25 additional datasets from the UCI repository [14].

5.2 Question 2

We divided each of the datasets into training and test sets for 10-fold cross validation. We discretized the data

Fig. 2. Effect of k on predictive accuracy.

TABLE 3

Accuracy of predicting correct value for 1 missing value.

Data Set	CNF	C4.5	MCV	Near. Neigh.
anneal	81.30±3.47	84.19±3.79	71.39±4.71 ●	81.79±2.42
audiology	94.72±3.41	96.03±2.48	91.15±5.06 ●	93.58±3.90
autos	89.67±5.59	84.29±7.80 ●	57.98±13.99 ●	85.11±6.58
balance-scale	3.36±2.31	33.15±7.17 ○	14.40±2.39 ○	4.48±0.97
breast-cancer	54.21±7.91	60.16±4.75 ○	44.40±8.97 ●	56.54±11.14
breast-w	55.80±3.97	56.08±3.80	49.93±3.66 ●	59.47±3.63 ○
colic	61.43±8.07	58.44±4.07	51.92±10.19 ●	32.85±9.17 ●
credit-a	66.09±6.70	70.29±5.89 ○	60.29±6.23 ●	64.73±4.57
credit-g	64.90±5.69	63.20±3.49	57.50±5.72 ●	59.38±4.47
diabetes	48.96±6.72	47.41±7.46	45.32±6.75 ●	49.36±5.78
glass	77.60±8.84	74.68±13.35	69.63±11.32 ●	77.51±8.56
heart-c	59.46±8.63	54.76±7.98	51.22±6.93 ●	59.46±7.15
heart-h	62.90±7.38	64.29±10.61	58.46±8.28 ●	53.53±13.65
heart-statlog	68.89±10.79	64.44±10.79 ●	65.19±8.04	57.76±6.38
hepatitis	73.63±11.12	77.33±8.43	70.21±10.61	49.27±12.89 ●
hypothyroid	91.15±1.46	92.36±1.91 ○	88.84±2.40 ●	90.35±2.37
iris	61.33±15.01	58.00±12.19	42.00±12.98 ●	57.67±7.21
kr-vs-kp	90.83±1.84	91.99±2.06 ○	80.79±2.55 ●	86.88±1.72
labor	68.00±20.26	64.33±15.40	58.00±18.14 ●	56.33±19.13
lymph	73.57±7.00	71.67±11.56	66.81±8.93 ●	73.06±7.82
mushroom	82.96±1.37	78.35±1.49 ●	60.56±1.62 ●	68.60±1.89
primary-tumor	75.22±9.62	79.35±7.21	74.04±7.45	78.32±7.76
segment	88.03±2.50	88.79±1.65	59.65±2.74 ●	89.79±2.20
sick	91.07±1.50	92.09±1.81 ○	88.28±2.24 ●	90.60±0.98
soybean	89.17±2.30	89.71±3.28	66.93±3.96 ●	79.04±4.77
vote	76.03±9.80	79.72±7.40	55.58±6.54 ●	56.33±7.65
vowel	75.05±3.72	68.89±3.04 ●	21.21±4.86 ●	72.66±4.92
yeast	74.86±5.46	66.77±3.19 ●	62.40±3.59	69.14±5.09
zoo	93.00±9.49	91.00±9.94	68.45±16.38 ●	88.09±9.22 ●

○, ● statistically significant improvement or degradation

after fold generation using unsupervised discretization into equiwidth buckets.

We compared our two algorithmic settings (class completion and multistream) to four algorithms available in the WEKA toolkit [15]: RIPPER (JRip), C4.5 (J48), PART and DecisionStump. All these algorithms follow a top-down approach for learning DNF-based rules or trees. We use DecisionStump as a baseline classifier. Since these algorithms can handle numeric data (or have built-in discretization), we also executed them on the non-discretized datasets. The default settings were used.

The results for the experiment are summarized in Table 1 and Table 2. The significance of differences was determined using a paired two sided t -test on the 0.05 level.

In these experiments the class completion setting performs significantly better than the multistream setting. The difference in accuracy can be explained by looking at the kind of rules that are learned in the two settings. In the

multistream setting, we learn rules of the form $(Class = c) \rightarrow \varphi$, where φ is a clause. In the class completion setting, this rule can be represented by the clause $\varphi \vee Class \neq c$. Additionally, however, in the class completion setting we learn clauses of the form $\varphi \vee Class = c$. These clauses provide additional expressive power in data with more than two classes. Indeed, in our experiments, we had 15 datasets with more than 2 classes, of which 9 have a significantly improved accuracy in the class completion setting, while in the other 15 datasets, we only observe 2 improvements. The p -value of Fisher's exact test on this contingency table is 0.021, which means that the correlation is significant with the usual 0.05 threshold. The differences between the two CNF approaches are not significant (according to a Sign test) if we restrict ourselves to the datasets with two classes.

Other differences between the algorithms are caused by the fact that clauses which are true for all classes, have an influence on the weighing scheme; also, for the same k , in the class completion approach clauses need to include the class attribute.

Compared to the other algorithms, our CNF class completion algorithm achieves a good predictive accuracy on most datasets, balancing wins and losses as compared to the other rule based learners RIPPER and PART.

5.3 Question 3

To answer this question we use a generalized version of our class completion setting, where all possible completions of missing attributes are scored. If two values score equally well, we choose the value which is most common in the training data.

We only use the discretized versions of the UCI datasets in these experiments. For every test example we mark M randomly chosen attributes of the original data as missing, where M is a parameter of the experiment. We choose not to delete class attributes. A prediction is deemed correct if the predicted values are *exactly* those that were originally in the example.

We ran experiments for $M = \{1, 3, 5\}$, where we compared the accuracy of our CNF algorithm with 3 alternative algorithms for missing value imputation. These algorithms are:

Most Common Value (MCV) which is a standard technique in missing value imputation that takes the most common value for the attribute. For $M > 1$ we selected the most common value for each attribute independently.

Nearest Neighbor For each test example we looked for the closest neighbor(s) in the training set using a Hamming distance¹. If multiple neighbors were found at the same distance we used voting to select a single completion.

C4.5 For each combination of attributes marked as missing in at least one example in the test data, we learned a decision tree or a set of decision trees on the training data using only non-missing attributes. In the single target (ST) setting we used M different trees to predict each missing

1. $dist(X, Y) = \sum_i^{atts} d(X_i, Y_i)$ with $d(a, a) = 0, d(a, ?) = 0.5, d(a, b) = 1$

TABLE 1
Experimental Results (1)

Data Set	CNF		C4.5		CBA	Default Acc
	cc5	ms5	non-disc	disc		
Data Set	(1)	(2)	(3)	(4)	(5)	(6)
anneal	96.66±1.96	91.09±2.61 ●	92.76±2.43 ●	85.19±2.94 ●	91.09±2.61 ●	76.17±0.54●
audiology	74.35±4.90	63.81±7.20 ●	76.94±6.31	76.94±6.31		25.20±1.74●
autos	80.52±11.75	78.55±10.86●	81.98±8.84	79.05±8.28	49.24±4.81 ●	32.69±2.34●
balance-scale	83.05±3.47	78.88±2.10 ●	64.50±3.45 ●	64.50±3.45 ●		45.76±0.56●
breast-cancer	70.25±9.23	73.76±6.05	73.42±3.34	73.42±3.34	66.40±7.15	70.30±1.43
breast-w	97.00±1.57	93.28±2.61 ●	94.71±2.43 ●	94.71±2.43 ●	95.42±1.75 ●	65.52±0.46●
colic	81.52±8.03	79.35±4.07	85.61±4.57	85.61±4.57	80.47±7.01	63.05±1.19●
credit-a	81.45±4.03	81.16±4.63	85.65±3.24 ○	84.49±2.27 ○	83.19±2.75	55.51±0.70●
credit-g	73.10±2.08	71.70±1.89 ●	70.90±5.09	71.60±4.27		70.00±0.00●
diabetes	72.28±4.09	70.57±4.60	73.83±4.95	70.07±4.01	71.23±2.82	65.11±0.36●
glass	61.73±7.91	56.62±11.86	65.95±11.55	61.75±9.82	57.55±7.86	35.54±2.69●
heart-c	82.85±9.24	80.88±9.59	77.27±5.90	77.89±5.17	79.87±8.29	54.45±1.35●
heart-h	80.53±8.57	80.54±9.31	80.52±10.64	80.17±10.38	81.21±11.15	63.95±1.43●
heart-statlog	78.89±5.25	80.74±4.88	80.00±7.45	79.26±6.10	79.26±5.26	55.56±0.00●
hepatitis	83.33±7.36	83.25±5.46	82.67±7.16	83.38±9.48	80.13±12.39	79.38±2.38
hypothyroid	93.24±0.52	92.68±0.57 ●	99.55±0.31 ○	93.29±0.53	92.26±0.11 ●	92.29±0.08●
iris	82.67±17.55	81.33±15.33	94.67±12.49○	93.33±10.42○	88.67±9.96	33.33±0.00●
kr-vs-kp	97.34±0.88	98.09±0.67 ○	99.50±0.37 ○	99.50±0.37 ○	81.04±1.28 ●	52.22±0.10●
labor	92.33±9.94	87.33±17.05	82.00±18.20	83.33±16.63●	91.00±12.38	64.67±3.22●
lymph	81.86±9.34	80.52±10.98	77.10±13.42	77.10±14.14	80.48±11.91	54.76±2.43●
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	99.95±0.09	51.80±0.05●
primary-tumor	42.78±4.92	38.66±8.25	43.96±5.65	43.96±5.65	35.99±7.04 ●	24.78±1.47●
segment	88.23±1.51	86.84±1.28 ●	96.54±0.82 ○	92.99±0.53 ○		14.29±0.00●
sick	97.16±0.70	96.71±0.84	98.78±0.70 ○	98.04±0.95 ○	93.85±0.11 ●	93.88±0.08●
soybean	94.58±2.09	90.63±3.03 ●	91.64±4.40 ●	91.64±4.40 ●		13.47±0.66●
vehicle	68.32±4.22	68.91±4.19	71.53±6.16	69.52±5.57		25.53±0.46●
vote	94.70±2.86	94.69±3.76	96.53±3.98	96.53±3.98	92.60±4.90	61.52±0.96●
vowel	95.25±1.78	91.92±1.78 ●	82.02±3.86 ●	76.46±4.29 ●	63.54±5.09 ●	9.09±0.00●
yeast	53.44±4.79	51.01±3.79 ●	55.66±4.82	52.29±3.47	51.75±3.42	31.20±0.30●
zoo	95.00±7.07	92.18±7.60	93.00±6.75	93.00±6.75	83.09±4.98 ●	40.55±1.72●
Win/Loss/Draw		1/11/18	6/5/19	5/6/19	0/9/15	0/28/2
Signed Wilcoxon		<0.001	0.624	0.484	<0.001	<0.001

○, ● statistically significant improvement or degradation

TABLE 2
Experimental Results (2)

Data Set	PART		RIPPER		DecisionStump	
	non-disc	disc	non-disc	disc	non-disc	disc
Data Set	(7)	(8)	(9)	(10)	(11)	(12)
anneal	94.43±2.83	88.08±3.07 ●	95.10±3.15	89.09±3.80 ●	77.17±3.97 ●	77.17±3.97 ●
audiology	80.47±5.76	80.47±5.76	75.26±5.02	75.26±5.02	46.44±2.66 ●	46.44±2.66 ●
autos	69.29±13.13●	68.81±10.17●	72.64±10.20●	73.67±9.31	45.43±7.66 ●	45.43±7.66 ●
balance-scale	78.08±2.38 ●	78.08±2.38 ●	71.86±4.56 ●	71.86±4.56 ●	54.39±2.47 ●	54.39±2.47 ●
breast-cancer	67.80±5.88	67.80±5.88	74.11±5.01	74.11±5.01	70.62±5.57	70.62±5.57
breast-w	94.14±3.33 ●	94.14±3.33 ●	93.56±2.74 ●	93.56±2.74 ●	88.27±4.30 ●	88.27±4.30 ●
colic	82.89±5.23	84.79±4.61	84.79±5.70	85.89±4.15	81.55±6.16	81.55±6.16
credit-a	83.77±2.72	85.07±3.49 ○	84.93±4.28 ○	85.36±3.01 ○	85.51±2.65 ○	85.51±2.65 ○
credit-g	70.30±4.52	70.50±2.84	72.70±3.86	70.30±3.30 ●	70.00±0.00 ●	70.00±0.00 ●
diabetes	74.09±4.61	71.35±4.94	73.70±3.98	69.53±5.24	72.14±5.05	66.79±4.27 ●
glass	69.22±7.35 ○	65.02±8.62	64.11±7.37	53.79±10.12●	44.39±2.26 ●	43.48±3.35 ●
heart-c	79.89±7.75	78.91±5.04	79.56±5.23	82.87±9.09	68.70±8.60 ●	68.70±8.60 ●
heart-h	80.53±8.70	81.21±9.06	77.44±10.93	81.18±9.20	81.90±7.69	81.90±7.69
heart-statlog	77.41±10.10	77.78±10.03	81.11±6.86	79.26±6.34	74.44±6.64	76.30±3.98
hepatitis	80.71±6.62	84.58±9.61	82.08±9.24	80.12±9.57	75.46±9.67 ●	82.04±8.26
hypothyroid	99.47±0.33 ○	93.37±0.64	99.34±0.40 ○	93.32±0.59	95.39±0.92 ○	93.43±0.61
iris	94.67±12.49○	93.33±10.42○	93.33±10.89○	93.33±8.31 ○	66.67±0.00 ●	66.00±2.11 ●
kr-vs-kp	99.34±0.31 ○	99.34±0.31 ○	99.12±0.73 ○	99.12±0.73 ○	66.05±1.31 ●	66.05±1.31 ●
labor	78.67±8.34 ●	82.00±17.44	84.00±15.70	82.00±14.67●	77.33±13.41●	77.33±13.41●
lymph	75.76±13.24	79.10±11.15	75.10±14.39	81.10±13.24	75.76±13.76	75.76±13.76
mushroom	100.00±0.00	100.00±0.00	100.00±0.00	100.00±0.00	88.68±1.13 ●	88.68±1.13 ●
primary-tumor	41.62±7.08	41.62±7.08	41.62±7.73	41.62±7.73	28.90±5.82 ●	28.90±5.82 ●
segment	96.45±1.41 ○	92.12±1.97 ○	95.84±1.82 ○	91.56±1.49 ○	28.53±0.14 ●	28.31±0.22 ●
sick	98.81±0.36 ○	97.64±1.06	98.49±0.43 ○	97.51±1.03	96.55±0.65	96.39±1.03 ●
soybean	91.21±4.54 ●	91.21±4.54 ●	91.50±3.40 ●	91.50±3.40 ●	27.96±2.64 ●	27.96±2.64 ●
vehicle	71.77±6.25	68.81±5.15	68.68±4.54	63.60±4.64	38.89±2.36 ●	39.12±3.29 ●
vote	96.07±4.63	96.07±4.63	95.84±3.59	95.84±3.59	95.84±3.59	95.84±3.59
vowel	79.80±5.26 ●	70.91±4.99 ●	71.11±5.11 ●	66.26±4.52 ●	17.37±1.15 ●	13.94±1.41 ●
yeast	54.98±4.72	50.47±4.15	58.89±4.12 ○	53.90±3.51	40.70±1.40 ●	34.57±1.68 ●
zoo	93.00±6.75	93.00±6.75	90.18±9.23	90.18±9.23	60.36±1.15 ●	60.36±1.15 ●
Win/Loss/Draw	6/6/18	4/6/20	7/5/18	4/8/18	2/20/8	1/21/8
Signed Wilcoxon	0.411	0.271	0.258	<0.159	<0.001	<0.001

○, ● statistically significant improvement or degradation

TABLE 4
Accuracy of predicting correct values for 3 missing values.

Data Set	CNF	random	C4.5 ST	MCV	Near. Neigh.	C4.5 MT
anneal	52.67±3.85	2.98±0.25●	56.24±4.94	37.65±6.29 ●	52.41±5.27	56.35±4.05
audiology	85.02±7.19	10.33±0.78●	85.83±4.14	76.11±5.16 ●	82.39±6.94	85.85±4.00
autos	77.10±5.47	1.56±0.31●	61.40±9.64 ●	17.10±9.41 ●	69.58±7.44 ●	61.95±9.00 ●
balance-scale	0.00±0.00	0.80±0.00○	1.60±1.30 ○	0.00±0.00	1.06±1.34 ○	0.00±0.00
breast-cancer	15.70±7.48	1.67±0.30●	14.72±5.95	9.05±6.36 ●	18.31±4.99	12.61±4.52
breast-w	25.61±4.33	0.10±0.00●	27.90±3.76 ○	26.04±4.97	31.40±5.61 ○	26.04±4.19
colic	30.72±7.94	3.06±1.44●	22.82±9.12 ●	16.30±8.04 ●	6.92±3.00 ●	20.40±7.51 ●
credit-a	30.72±4.92	3.49±0.53●	32.46±4.11 ○	24.06±4.33 ●	27.65±3.83	30.00±6.83
credit-g	27.30±4.50	1.86±0.20●	23.70±3.80 ●	19.00±3.37 ●	20.42±4.62 ●	23.70±3.80 ●
diabetes	10.30±4.76	0.60±0.13●	10.03±3.98	10.82±4.46	9.99±3.85	10.30±3.91
glass	49.55±13.42	1.57±0.84●	45.32±10.95	38.35±11.47●	42.09±8.56	47.21±11.46
heart-c	20.11±8.72	3.02±0.56●	17.51±8.31	14.55±5.29	18.70±6.74	15.88±6.87
heart-h	18.78±7.90	2.59±0.67●	19.46±10.25	19.82±8.86	17.34±7.50	19.77±7.12
heart-statlog	28.52±7.62	3.95±1.64●	25.19±7.77	23.33±11.85	26.79±6.97	22.96±10.88
hepatitis	37.50±11.70	7.59±0.79●	40.13±11.01	37.33±9.22	19.66±6.52 ●	40.08±11.65
hypothyroid	72.53±2.24	11.15±0.79●	76.62±4.62 ○	68.24±4.36 ●	74.42±2.85 ○	77.01±4.40 ○
iris	14.67±9.84	0.29±0.14●	17.33±11.84	9.33±8.43 ●	27.74±13.00○	14.67±10.33
kr-vs-kp	75.72±1.38	12.16±0.06●	77.57±1.89 ○	54.10±1.80 ●	65.77±2.92 ●	76.60±1.65 ○
labor	44.00±21.19	8.09±1.84●	38.33±24.10	26.33±17.17	21.58±22.30●	41.00±30.71
lymph	35.86±10.60	5.66±0.63●	29.76±9.63	33.00±11.69	31.87±11.74	28.33±9.71
mushroom	54.73±1.32	1.42±0.05●	47.12±0.80 ●	25.00±1.86 ●	32.00±1.91 ●	46.05±0.95 ●
primary-tumor	44.55±11.81	10.64±0.49●	50.12±9.88	43.33±7.45	48.40±9.83	49.88±10.87
segment	68.36±3.20	1.26±0.28●	71.69±2.01 ○	24.76±2.80 ●	70.49±3.60	71.21±2.80 ○
sick	73.60±2.53	11.23±0.87●	76.96±1.91 ○	67.92±3.31 ●	75.51±1.76	76.88±2.51 ○
soybean	67.36±3.56	5.78±0.41●	70.73±5.77 ○	31.62±4.64 ●	63.85±4.70	69.10±5.47
vote	44.69±5.21	12.91±0.86●	53.22±6.52 ○	28.58±7.99 ●	34.09±6.29 ●	51.14±5.11 ○
vowel	41.11±2.90	0.24±0.04●	32.63±3.56 ●	2.02±1.26 ●	38.07±5.87	38.99±3.20
yeast	37.60±6.56	1.08±0.15●	28.02±4.29 ●	21.50±4.83 ●	33.54±4.62 ●	26.61±4.60 ●
zoo	69.18±22.44	11.01±1.57●	64.27±21.81	36.64±9.43 ●	74.52±10.87	65.27±20.23

○, ● statistically significant improvement or degradation

attribute separately. In the multi target (MT) setting we combined the M missing attributes into one and used a single tree to predict all M missing values at the same time. Tables 3 and 4 show the results for $M=1$ and $M=3$.

For one missing value, our results show clearly significant improvements compared to the MCV and the nearest neighbor algorithm. The difference with C4.5 is insignificant.

For higher numbers of missing values, the CNF approach is still significantly better than MCV and nearest neighbor, according to a sign test, but less significantly so. Compared to both approaches based on C4.5 we do not observe significant improvements, although it should be pointed out that our method is more efficient during training, as we do not need to compute a decision tree for all occurring subsets of missing attributes.

6 CONCLUSIONS AND FUTURE DIRECTIONS

Starting from the work of Mitchell on boundary sets and of Valiant on the learnability of k -CNF formulae, we developed an algorithm for mining the true k -CNF formulae. This algorithm forms an alternative to existing rule-learning systems in that it is incremental, performs one pass through the data, has only one parameter and can be used for imputing missing values. Experiments showed that the approach is competitive with state-of-the-art learners, while possessing different characteristics. The approach also puts the results of Mitchell and Valiant, which are mainly used for theoretical purposes, in a new, more practical perspective.

ACKNOWLEDGEMENTS

This work was supported by a Postdoc grant from the Research Foundation — Flanders. This work is partially supported by the GOA project 2008/08 Probabilistic Logic Learning. This work uses HPC resources <http://ludit.kuleuven.be/hpc>.

REFERENCES

- [1] W. W. Cohen, "Fast effective rule induction," in *ICML*, 1995, pp. 115–123.
- [2] E. Frank and I. H. Witten, "Generating accurate rule sets without global optimization," in *ICML*, 1998, pp. 144–151.
- [3] X. Yin and J. Han, "CPAR: Classification based on predictive association rules," in *SDM*, 2003, pp. 331–335.
- [4] J. R. Quinlan, *C4.5: programs for machine learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.
- [5] B. Liu, W. Hsu, and Y. Ma, "Integrating classification and association rule mining," in *KDD*, 1998, pp. 80–86.
- [6] W. Li, J. Han, and J. Pei, "CMAR: Accurate and efficient classification based on multiple class-association rules," in *ICDM*, 2001, pp. 369–376.
- [7] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
- [8] T. M. Mitchell, "Version spaces: A candidate elimination approach to rule learning," in *Fifth International Joint Conference on Artificial Intelligence*, 1977, pp. 305–310.
- [9] R. J. Mooney, "Encouraging experimental results on learning cnf," *Machine Learning*, vol. 19, pp. 79–92, 1995.
- [10] W. Sverdluk and R. G. Reynolds, "Dynamic version spaces in machine learning," in *ICTAI*, 1992, pp. 308–315.
- [11] T.-P. Hong and S.-S. Tseng, "A generalized version space learning algorithm for noisy and uncertain data," *IEEE Trans. Knowl. Data Eng.*, vol. 9, no. 2, pp. 336–340, 1997.
- [12] R. J. Bayardo Jr., "Efficiently mining long patterns from databases," in *SIGMOD Conference*, 1998, pp. 85–93.
- [13] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo, "Fast discovery of association rules," in *Advances in Knowledge Discovery and Data Mining*. AAAI/MIT Press, 1996, pp. 307–328.
- [14] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: <http://www.ics.uci.edu/~mllearn/MLRepository.html>
- [15] I. H. Witten and E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd ed. Morgan Kaufmann, 2005.