MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Mining Features for Sequence Classification

Neal Lesh, Mohammed J. Zaki, Mitsunori Ogihara

TR98-22 December 1998

Abstract

Classification algorithms are difficult to apply to sequential examples, such as plan executions or text, because there is a vast number of potentially useful features for describing each example. Past work on feature selection has focused on searching the space of all subsets of the available features which is intractable for large feature sets. We adapt data mining techniques to act as a preprocessor to select features for standard classification algorithms such as Naive Bayes and Winnow. We apply our algorithm to the task of predicting whether or not a plan will succeed or fail, during plan execution. The features produced by our algorithm improve classification accuracy by 10-50 percent in our experiments.

MERL – A MITSUBISHI ELECTRIC RESEARCH LABORATORY
http://www.merl.com

Mining features for sequence classification

Neal Lesh Mohammed J. Zaki*
Mitsunori Ogihara†

TR-98-22 December 1998

Abstract

Classification algorithms are difficult to apply to sequential examples, such as plan executions or text, because there is a vast number of potentially useful features for describing each example. Past work on feature selection has focused on searching the space of all subsets of the available features which is intractable for large feature sets. We adapt data mining techniques to act as a preprocessor to select features for standard classification algorithms such as Naive Bayes and Winnow. We apply our algorithm to the task of predicting whether or not a plan will succeed or fail, during plan execution. The features produced by our algorithm improve classification accuracy by 10-50% in our experiments.

Submitted to IJCAI'99.

*Rensselaer Polytechnic Institute
†University of Rochester

Publication History:–

1. First printing, TR-98-22, December 1998

1 Introduction

Some classification algorithms have been shown to work well when there are thousands of features for describing each example (e.g, [14; 16]). In some domains, however, the number of potentially useful features is exponential in the size of the examples. Data mining algorithms (e.g., [1]) have been used to search through billions of rules, or patterns, and select the most interesting ones. In this paper, we adapt data mining techniques to act as a preprocessor to construct a set of features to use for classification.

This work was motivated by the task of plan monitoring in stochastic domains (e.g, [17]). Probabilistic planners (e.g. [12]) construct plans with high probability of achieving their goal. The task of monitoring is to "watch" as the plan is executed and predict, in advance, whether the plan will most likely succeed or fail to facilitate re-planning.

There are three steps in our approach. First, we simulate the plan repeatedly to generate execution traces. Probabilistic planners use a domain model that indicates the probability of state s_i resulting from executing action a_j in state s_k for all states and actions. We use this information to perform Monte Carlo simulation. Second, we mine the simulated execution traces for features that are common in either successful or unsuccessful execution traces, but not in both. Finally, we train a classifier to predict success or failure on a second set of simulated traces, using the features produced in the second step.

Plan monitoring (or monitoring any probabilistic process that we can simulate) is an attractive area for machine learning because there is an essentially unlimited supply of training data. Although we cannot consider all possible execution paths, because the number of paths is exponential in the length of the plan, or process, we can generate arbitrary numbers of new examples with Monte Carlo simulation. The problem of over-fitting is reduced because we can test our hypotheses on "fresh" data sets.

The contribution of this paper is to combine two powerful paradigms for classification: sequence mining which can efficiently search for patterns that are correlated with the target classes, and classification algorithms which learn to weigh evidence from different features to classify new examples. Additionally, we present a criteria for selecting features, and present pruning rules that allow for more efficient mining of the features.

The rest of this paper is organized as follows. First, we demonstrate the main ideas described in this paper with a simple example. We then formulate and present a solution for the feature mining problem. We then describe our experiments and conclude with discussion of related and future work.

2 Example: poker

We now preview the main ideas in this paper with a simple, illustrative example.

Suppose we observe three people playing poker with betting sequences and outcomes such as:

example: P_1 Bets 3, P_2 Calls, P_3 Raises 2, P_1 Raises 1, P_2 Folds, P_3 Calls $\Rightarrow$ P_1 wins

Our objective is to learn a function that predicts who is most likely to win given a betting sequence. This task resembles standard classification: we are given labelled training examples and must produce a function that classifies new, unlabelled examples. Classifiers require, however, that examples be represented as vectors of feature-value pairs. This paper addresses the problem of selecting features to represent the betting sequences.

First, consider an obvious, but poor, feature set. Let N be the length of the longest betting sequence. We can represent betting sequences with $3N$ features by generating a distinct feature for every $0 \leq i \leq N$, for the person who made the ith bet, for the type of the ith bet, and for the amount of the ith bet. In section 4, we show experimentally that this feature set leads to poor classification. One problem with these features is that they do not express that some *sequence* of events took place. Consider the following features:

Feature: P_1 raises twice
Feature: P_2 folds and then P_1 raises 2 dollars

The first feature would be important if, for example, P_1 tends to win whenever she raises twice. A classifier could construct a boolean expression out of the features described above to capture the notion "P_1 raises twice", but the expression would have N^2 disjuncts.[1]

An alternative is to use a much larger feature set. If there are 3 players, 4 bids, and 5 different amounts then there are $4 \times 5 \times 6 = 120$ partial specifications of a bet, such as "someone bets 3". We can chain partial specifications together with an "and then" relation, as in "P_1 raises and then someone bets 3". The number of such features of length K is 120^K. The problem with this feature set is that it is too large. Sets of 10,000 features are considered large [5]. Furthermore, irrelevant or redundant features can reduce classification accuracy [4].

We adopt a middle ground between these two extremes. We use data mining techniques to search through the second, huge feature set and select a subset. We show that a criteria similar to that used in the general knowledge discovery task works well for deciding which features will be useful.

2.1 Analyzing probabilistic processes

What does this example have to do with plan monitoring? Suppose that instead of watching people play cards, we are given probabilistic rules which describe how the players bet, such as "if player P_1's hand cannot beat two pair, then P_1 will fold with .75 probability, or otherwise call." Analytically computing which player is most likely

[1] It would need a disjunct for "P_1 raises in the ith bet and in the jth" for all $i, j < N$ where $i \neq j$.

to win given a betting sequence can be intractable. Instead, we can perform Monte Carlo simulation to generate betting sequences and treat these sequences as training examples, as described above.

For plan monitoring, we are given a probabilistic plan P, goal G, and domain description D. We assume that some or all of the actions in P have an observable outcome. The task of plan monitoring is to predict whether the plan will succeed or fail given the observed results of the partial execution of plan P. Computing the exact probability that the plan will succeed is exponential in the length of P [13]. As a more tractable alternative, we can repeatedly simulate plan P and label each simulation by whether or not goal G is true in the final state. Since we want predict the final outcome of the plan in advance, we can train a classifier with, say, only the first 10 time steps of each simulation. During plan execution, this classifier can be used as a plan monitor by feeding the first ten steps of the real execution into the classifier. The monitor's output will be an indication of whether the plan is likely to succeed or fail.

Note that the plan monitor is trained prior to execution. With the classifiers we use, the time required during plan execution is linear in the number of features.

3 Data mining for features

We now formulate and present an algorithm for feature mining. We begin by adopting the following terminology, which closely resembles that used for sequence mining.

Let $\mathcal{F}$ be a set of distinct features, each with some finite set of possible values. Let $\mathcal{I}$ contain a unique element for every possible feature-value pair. A *sequence* is an ordered list of subsets of $\mathcal{I}$. A sequence α is denoted as $(\alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_n)$ where each sequence element α_i is a subset of $\mathcal{I}$. The *length* of sequence $(\alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_n)$ is n and its *width* is the maximum size of any α_i for $1 \leq i \leq n$. We say that α is a subsequence of β, denoted as $\alpha \prec \beta$, if there exists integers $i_1 < i_2 < ... < i_n$ such that $\alpha_j \subseteq \beta_{i_j}$ for all α_j.

Let $\mathcal{C}$ be a set of class labels. An *example* is a pair $\langle \alpha, c \rangle$ where α is a sequence and $c \in \mathcal{C}$ is a label. An example $\langle \alpha, c \rangle$ is said to *contain* sequence β if $\beta \prec \alpha$.

Let $\mathcal{D}$ be a set of examples. The *frequency* of sequence β in $\mathcal{D}$, denoted $fr(\beta, \mathcal{D})$, is the fraction of examples in $\mathcal{D}$ that contain β. Let β be a sequence and c be a class label. The *confidence* of the rule $\beta \Rightarrow c$, denoted $conf(\beta, c, \mathcal{D})$, is the conditional probability that c is the label of an example in $\mathcal{D}$ given that it contains sequence β. That is, $conf(\beta, c, \mathcal{D}) = \frac{fr(\beta, \mathcal{D}_c)}{fr(\beta, \mathcal{D})}$ where $\mathcal{D}_c$ is the subset of examples in $\mathcal{D}$ with class label c.

A *sequence classifier* is a function from sequences to $\mathcal{C} \cup \{null\}$. A classifier can be evaluated using standard metrics such as accuracy and coverage.

Finally, we describe how sequences $\beta_1, ..., \beta_n$ can be used as features for classification. Recall that the input to a standard classifier is an example represented as vector of feature-value pairs. We represent a sequence α as a vector of feature-value pairs by treating each sequence β_i as a boolean feature that is true iff $\beta_i \preceq \alpha$. For example, suppose the features are $f_1 = A \rightarrow B$, $f_2 = A \rightarrow BC$, and $f_3 = AC$. The sequence $AB \rightarrow BC$ would be represented as $\langle f_1, true \rangle, \langle f_2, true \rangle, \langle f_3, false \rangle$. The sequence ABC $\rightarrow$ B would be represented as $\langle f_1, true \rangle, \langle f_2, false \rangle, \langle f_3, true \rangle$.

3.1 Selection criteria for mining

We now specify our selection criteria for selecting features to use for classification. Our objective is to find sequences such that representing examples with these sequences will yield a highly accurate sequence classifier. Certainly, the criteria for selecting features might depend on the domain and the classifier being used. We believe, however, that the following domain-and-classifier-independent heuristics are useful for selecting sequences to serve as features:

1. Features should be frequent.
2. Features should be distinctive of at least one class.
3. Feature sets should not contain redundant features.

The intuition behind the first heuristic is simply that rare features can, by definition, only rarely be useful for classifying examples. In our problem formulation, this heuristic translates into a requirement that all features have some minimum frequency in the training set.

The intuition for the second heuristic is that features that are equally likely in all classes do not help determine which class an example belongs to. Of course, a conjunction of multiple non-distinctive features can be distinctive. In this case, our algorithm prefers to use the distinctive conjunction as a feature rather than the non-distinctive conjuncts. We encode this heuristic by requiring that each selected feature be significantly correlated with at least one class that it is frequent in.

The motivation for our third heuristic is that if two features are closely correlated with each other, then either of them is as useful for classification as both are. We show below that we can reduce the number of features and the time needed to mine for features by pruning redundant rules. In addition to wanting to prune features which provide the same information, we also want to prune a feature if there is another feature available that provides strictly more information. Let $M(f, \mathcal{D})$ be the set of examples in $\mathcal{D}$ that contain feature f. We say that feature f_1 *subsumes* feature f_2 with respect to predicting class c in data set $\mathcal{D}$ iff $M(f_2, \mathcal{D}_c) \subseteq M(f_1, \mathcal{D}_c)$ and $M(f_1, \mathcal{D}_{\neg c}) \subseteq M(f_2, \mathcal{D}_{\neg c})$

We can now define the feature mining task. The inputs to the FEATUREMINE algorithm are a set of examples $\mathcal{D}$ and parameters min_{fr}, max_w, and max_l. The output is a non-redundant set of the frequent and distinctive features of width max_w and length max_l. Formally:
Feature mining: Given examples $\mathcal{D}$ and parameters min_{fr}, max_w, and max_l return feature set $\mathcal{F}$ such that for every feature f_i and every class $c_j \in \mathcal{C}$, if $length(f_i) \leq max_l$ and $width(f_i) \leq max_w$ and $fr(\beta, \mathcal{D}_{c_j}) \geq min_{fr}$

and $conf(\beta, c_j, \mathcal{D})$ is significantly greater than $|\mathcal{D}_c|/|\mathcal{D}|$ then $\mathcal{F}$ contains f_i or contains a feature that subsumes f_i with respect to class c_j in data set $\mathcal{D}$.[2]

3.2 Efficient mining of features

We now present the FEATUREMINE algorithm which leverages existing data mining techniques to efficiently mine features from a set of training examples.

Sequence mining algorithms are designed to discover highly frequent and confident patterns in sequential data sets and so are well suited to our task. FEATUREMINE is based on the recently proposed UNNAMED algorithm (name and citation omitted for blind review) for fast discovery of sequential patterns. The algorithm uses the observation that the subsequence relation $\preceq$ defines a partial order on sequences. If $\alpha \prec \beta$, we say that α is *more general than* β, or β is *more specific than* α. The relation $\preceq$ is a *monotone specialization relation* with respect to the frequency $fr(\alpha, \mathcal{D})$, i.e., if β is a frequent sequence, then all subsequences $\alpha \preceq \beta$ are also frequent.

The UNNAMED algorithm systematically searches the sequence lattice spanned by the subsequence relation, from general to specific sequences, in a breadth/depth-first manner. UNNAMED starts with sequences containing a single item. During each iteration, the sequences from the previous iteration that have frequency min_{fr} or greater are extended by one more item. The new item is chosen from an equivalence class of k length sequences which share the same $k-1$ length suffix. For example, if suffix equivalence class $[C]$ has the elements $A \mapsto C$, and $B \mapsto C$, then the only possible frequent sequences at the next step can be $A \mapsto B \mapsto C$, $B \mapsto A \mapsto C$, and $(AB) \mapsto C$. No other item X can lead to a frequent sequence with the suffix C, unless (XC) or $X \mapsto C$ is also in $[C]$. This use of equivalence classes decomposes the original search space into smaller sub-spaces, which can be solved independently, and results in efficient enumeration of all the frequent patterns.

To construct FEATUREMINE, we adapted the UNNAMED algorithm to search data bases of labelled examples. FEATUREMINE mines the patterns predictive of all the classes in the database, simultaneously. As opposed to previous approaches that first mine millions of patterns and then apply pruning as a post-processing step, FEATUREMINE integrates pruning techniques in the mining algorithm itself.

The first pruning rule is that we do not extend (i.e, specialize) any feature with 100% accuracy. Let f_1 be a feature contained by examples of only one class. Specializations of f_1 may pass the frequency and confidence tests in the definition of feature mining, but will be subsumed by f_1. The following Lemma, which follows from the definition of subsume, justifies this pruning rule:

Lemma 1: If $f_i \prec f_j$ and $conf(f_i, c, \mathcal{D}) = 1.0$ then f_i subsumes f_j with respect to class c.

Our next pruning rule concerns correlations between individual items. Recall that the examples in $\mathcal{D}$ are represented as a sequence of sets. We say that $A \Rightarrow B$ in examples $\mathcal{D}$ if B occurs in every set in every sequence in $\mathcal{D}$ in which A occurs. The following lemma states that if $A \Rightarrow B$ then any feature containing a set with both A and B will be subsumed by one of its generalizations:

Lemma 2: Let $\alpha = \alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_n$ where $A, B \in \alpha_i$ for some $1 \leq i \leq n$. If $A \Rightarrow B$, then α will be subsumed by $\alpha_1 \rightarrow ...\alpha_{i-1} \rightarrow \alpha_i - B \rightarrow \alpha_{i+1}... \rightarrow \alpha_n$.

We precompute the set of all $A \Rightarrow B$ relations and immediately prune any feature, during the search, that contains a set with both A and B. In section 4, we discuss why $A \Rightarrow B$ relations arise and show they are crucial for the success of our approach for some problems.

4 Empirical evaluation

We now describe experiments to test whether the features produced by our system improve the performance of the Winnow [14] and Naive Bayes [7] classification algorithms.

Winnow is a multiplicative weight-updating algorithm. We used a variant of Winnow that maintains a weight $w_{i,j}$ for each feature f_i and class c_j. Given an example, the activation level for class c_j is $\sum_{i=0}^{n} w_{i,j} x_i$ where x_i is 1 if feature f_i is true in the example, or 0 otherwise. Given an example, Winnow outputs the class with the highest activation level. During training, Winnow iterates through the training examples. If Winnow's classification of a training example does not agree with its label then Winnow updates the weights of each feature f_i that was true in the example: it multiplies the weights for the correct class by some constant $\alpha > 1$ and multiples the weights for the incorrect classes by some constant $\beta < 1$. In our experiments, $\alpha = 1.1$ and $\beta = .91$

For each feature f_i and class c_j, Naive Bayes computes $\mathrm{P}(f_i|c_j)$ as the fraction of training examples of class c_j that contain f_i. Given a new example in which features $f_1, ...f_n$ are true, Naive Bayes returns the class that maximizes $P(c_j) \times P(f_1|c_j) \times ... \times P(f_n|c_j)$. Even though the Naive Bayes algorithm appears to make the unjustified assumption that all features are independent it has been shown to perform surprisingly well, often doing as well as or better than C4.5 [6].

We now describe the domains we tested our approach in and then discuss the results of our experiments.

Random parity problems: We first describe a (non-sequential) problem designed to demonstrate the potential value of our approach.

Each problem is defined by three integers, N, M, and L. For every $0 \leq i \leq N$ and $0 \leq j \leq M$, there is a boolean feature $f_{i,j}$. Additionally, for $0 \leq k \leq L$, there is an irrelevant, boolean feature f_k. Each instance is set containing a feature/boolean pair for each feature and its value in that instance. Thus, there are $2^{N \times M + L}$ distinct instances. Each feature has a 50% chance of being true and thus every instance is equally likely.

We also choose N weights $w_1, ..., w_N$, and assign each instance one of two class labels (`ON` or `OFF`) as follows. The "score" of an instance is the sum of the weights w_i

[2] We use a chi-squared test to determine significance.

Experiment	Winnow	Winnow with times × features	Winnow with FeatureMine	Bayes	Bayes with time × features	Bayes with FeatureMine
parity, $N = 5, M = 3, L = 5$	.51	N/A	.96	.50	N/A	.96
parity, $N = 3, M = 4, L = 8$	.50	N/A	.98	.50	N/A	1.0
parity, $N = 10, M = 4, L = 10$	.50	N/A	.88	.50	N/A	.84
fire, time = 5	.60	.65	.79	.69	.71	.81
fire, time = 10	.56	.77	.85	.68	.68	.75
fire, time = 15	.52	.75	.88	.68	.68	.72
spelling, their vs. there	.70	N/A	.94	.75	N/A	.78
spelling, I vs. me	.86	N/A	.94	.66	N/A	.90
spelling, than vs. then	.83	N/A	.92	.79	N/A	.81
spelling, you're vs. your	.77	N/A	.86	.77	N/A	.86

Table 1: Classification results: the average classification accuracy using different feature sets to represent the examples. The highest accuracy was obtained with the features produced by the FeatureMine algorithm

for which the parity of features $f_{i,1}, ...f_{i,M}$ is even. If an instance's score is greater than half the sum of all the weights, $\sum_{i=1}^{N} w_i$, then the instance is assigned class label `ON`, otherwise it is assigned `OFF`. Example features produced by FeatureMine are ($f_{1,1}$=true, $f_{1,2}$=true), and ($f_{4,1}$=true, $f_{4,2}$=false,$f_{4,3}$ =true).

In the experiments reported here, we used a min_{fr} of .02 to .05, $max_w = 4$, and $max_l = 1$.

Forest fire plans: We constructed a simple forest-fire domain based loosely on the Phoenix fire simulator [9].[3]

We use a grid representation of the terrain. Each grid cell can contain vegetation, water, or a base. At the beginning of each simulation, the fire is started at a random location. In each iteration of the simulation, the fire spreads stochastically. The probability of a cell igniting at time t is calculated based on the cell's vegetation, the wind direction, and how many of the cell's neighbors are burning at time $t-1$. Additionally, bulldozers are used to contain the fire. For each example terrain, we hand-designed a plan for bulldozers to dig a fire line to stop the fire. The bulldozer's speed varies from simulation to simulation. An example simulation looks like:

(time0 Ignite X3 Y7), (time0 MoveTo BD1 X3 Y4),
(time0 MoveTo BD2 X7 Y4), (time0 DigAt BD2 X7 Y4),
.... (time8 DigAt BD2 X7 Y3), (time8 Ignite X3 Y9),
(time8 DigAt BD1 X2 Y3), (time8 Ignite X5 Y8),
... (time32 Ignite X6 Y1), (time32 Ignite X6 Y0), ...

We tag each plan with `SUCCESS` if none of the locations with bases have been burned in the final state, or `FAILURE` otherwise. To train a plan monitor that can predict at time k whether or not the bases will ultimately be burned, we only include events which occur by time k in the training examples. Example features produced by FeatureMine in this domain are:

(MoveTo BD1 X2) → (time6), and
(Ignite X2) → (time8 MoveTo Y3))

The first sequence holds if bulldozer BD1 moves to the second column before time 6. The second holds if a fire ignites anywhere in the second column and then any bulldozer moves to the third row at time 8.

[3]Execution traces are available by email from the authors.

Many correlations used by our second pruning rule described in section 3.2 arise in these data sets. For example, Y8 ⇒ Ignite arises in one of our test plans in which a bulldozer never moves in the eighth column.

In the experiments reported here, we used a $min_{fr} = .2$, $max_w = 3$, and $max_l = 3$.

Context-sensitive spelling correction

We also tested our algorithm on the task of correcting spelling errors that result in valid words, such as substituting *there* for *their* ([8]). For each test, we chose two commonly confused words and searched for sentences in the 1-million-word Brown corpus [10] containing either word. We removed the target word and then represented each word by the word itself, the part-of-speech tag in the Brown corpus, and the position relative to the target word. For example, the sentence "And then there is politics" is translated into (word=and tag=cc pos=-2) → (word=then tag=rb pos=-1) → (word=is tag=bez pos=+1) → (word=politics tag=nn pos=+2).

Example features produced by FeatureMine include (pos=+3) → (word=the), indicating that the word *the* occurs at least 3 words after the target word, and (pos=-4) → (tag=nn) → (pos=+1), indicating that a noun occurs within three words before the target word.

In the experiments reported here, we used a $min_{fr} = .05$, $max_w = 3$, and $max_l = 2$.

4.1 Results

For each test in the parity and fire domains, we generated 7,000 random training examples. We mined features from 1,000 examples, pruned features that did not pass a chi-squared significance test (for correlation to a class the feature was frequent in) in 2,000 examples, and trained the classifier on the remaining 5,000 examples. We then tested on 1,000 additional examples. The results in Tables 1 and 2 the tables are averaged over 25-50 such tests. For the spelling correction, we used all the examples in the Brown corpus, roughly 1000-4000 examples per word set, split 80-20 (by sentence) into training and test sets. We mined features from 500 sentences and trained the classifier on the entire training set.

Table 1 shows that the features produced by FeatureMine improved classification performance. We compared using the feature set produced by FeatureM-

Experiment	Evaluated features	Selected features
random, $N = 10, M = 4, L = 10$	7,693,200	196
fire world, time =10	64,766	553
spelling, there vs. their	782,264	318

Table 2: Mining results: number of features considered and returned by FEATUREMINE

INE with using only the primitive features themselves, i.e. features of length 1. In the fire domain, we also evaluated the feature set containing a feature for each primitive feature at each time step.[4] Both Winnow and Naive Bayes performed much better with the features produced by FEATUREMINE. In the parity experiments, the mined features dramatically improved the performance of the classifiers and in the other experiments the mined features improved the accuracy of the classifiers by a significant amount, often more than 20%.

Table 2 shows the number of features evaluated and the number returned, for several of the problems. For the largest random parity problem, FEATUREMINE evaluated more than 7 million features and selected only about 200. There were in fact 100 million possible features[5] but most of were rejected implicitly by the pruning rules.

Table 3 shows the impact of the $A \Rightarrow B$ pruning rule described in Section 3.2 on mining time. The results are from one data set from each domain, with slightly higher values for max_l and max_w than in the above experiments. The pruning rule did not improve mining time in all cases, but made a tremendous difference in the fire world problems, where the same event descriptors often appear together. Without $A \Rightarrow B$ pruning, the fire world problems are essentially unsolvable because FEATUREMINE finds over 20 million frequent sequences.

5 Related work

A great deal of work has been done on feature-subset selection, motivated by the observation that classifiers can perform worse with feature set $\mathcal{F}$ than with some $\mathcal{F}' \subset \mathcal{F}$ (e.g., [4]). The algorithms explore the exponentially large space of all subsets of a given feature set. In contrast, we explore exponentially large sets of potential features, but evaluate each feature independently. The feature-subset approach seems infeasible for the problems we consider, which contain hundreds of thousands to millions of potential features.

[8] applied a Winnow-based algorithm to context-sensitive spelling correction. They use sets of 10,000 to 40,000 features and either use all of these features or prune some based on the classification accuracy of the individual features. They obtain higher accuracy than we did. Their approach, however, involves an ensemble of Winnows, combined by majority weighting, and they took more care in choosing good parameters for this specific task. Our goal, here, is to demonstrate that the features produced by FEATUREMINE improve classification performance.

Data mining algorithms have often been applied to the task of classification. [15] build decision lists out of patterns found by association mining. [2] and [3] both combine association rules to form classifiers. Our use of sequence mining is a generalization on association mining. (In the random parity problems, sequence mining reduces to association mining.) Additionally, while previous work has explored new methods for combining association rules to build classifiers, the thrust of our work has been to leverage and augment standard classification algorithms. Our pruning rules resemble ones used by [19], which also employs data mining techniques to construct decision lists. Previous work on using data mining for classification has focused on combining highly accurate rules together. By contrast, our classification algorithms can weigh evidence from many features which each have low accuracy in order to classify new examples.

Our work is close in spirit to [11], which also constructs a set of sequential, boolean features for use by classification algorithms. They employ a heuristic search algorithm, called FGEN, which incrementally generalizes features to cover more and more of the training examples, based on its classification performance on a hold-out set of training data, whereas we perform an exhaustive search (to some depth) and accept all features which meet our selection criteria. Additionally, we use a different feature language and have tested our approaches on different classifiers then they have.

Past work has applied data mining to plans. [13; 20] mine plan traces in order to analyze and improve plans, but do make use of the pruning rules we have presented here. [18] applies data mining techniques to plans in order to infer causal, probabilistic models of plan operators. These systems do no use patterns from data mining as features for standard classification algorithms.

6 Future work and conclusions

We plan to test our approach on other data sets, multi-class problems, and other classification algorithms. We also intend to explore methods for finding features that will most improve classification. One idea is to split the training data into examples that the classifier correctly and incorrectly classifies and then mine for features that distinguish between these two sets.

In this paper, we have adapted data mining techniques to act as a preprocessor to construct features for use by standard classification algorithms. We search through millions of features to select a few hundred or thousand features that are frequent and distinctive. We applied our algorithm, FEATUREMINE, to the tasks of spelling correction and classifying simulated plan traces. The features produced by our algorithm FEATUREMINE significantly improve classification accuracy.

[4] This is the feature set of size $3N$ described in section 2.

[5] There are 50 booleans features, giving rise to 100 feature-value pairs. We searched to depth 4 (because $M = 4$).

Experiment	CPU seconds with no pruning	CPU seconds with only $A \Rightarrow B$ pruning	CPU seconds with all pruning	Features examined with no pruning	Features examined with only $A \Rightarrow B$ pruning	Features examined with all pruning
random	320	337	337	1,547,122	1,547,122	1,547,122
fire world	5.8 *hours*	560	559	25,336,097	511,215	511,215
spelling	490	407	410	1,126,114	999,327	971,085

Table 3: Impact of pruning rules: running time and nodes visited for FEATUREMINE with and without the $A \Rightarrow B$ pruning. Results taken from one data set for each example.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In *Intl. Conf. on Data Engg.*, 1995.

[2] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *Proc. 3th Int. Conf. on KDD*, pages 115–118, 1997.

[3] R.J. Jr. Bayardo. Brute-force mining of high-confidence classification rules. In *Proc. 3rd Int. Conf. on KDD*, pages 123–126, 1997.

[4] R. Caruana and D. Freitag. Greedy attribute selection. In *Proc. 11th Int. Conf. Machine Learning*, pages 28–36, 1994.

[5] T. Dietterich. Machine-learning research. *AI Magazine*, pages 97–136, 1997.

[6] P. Domingos and M. Pazzani. Beyond independence: conditions for the optimality of the simple bayesian classifier. In *Proc. 13th Int. Conf. Machine Learning*, pages 105–112, 1996.

[7] R.O. Duda and P.E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[8] A. Golding and D. Roth. Applying winnow to context-sensitive spelling correction. In *Proc. 13th Int. Conf. Machine Learning*, pages 180–190, 1996.

[9] D.M Hart and P.R Cohen. Predicting and explaining success and task duration in the phoenix planner. In *Proceedings of the First International Conference on AI Planning Systems*, pages 106–115, 1992.

[10] H. Kucera and W.N. Francis. *Computational Analysis of Present-Day American English*. Brown University Press, Providence, RI, 1967.

[11] D. Kudenko and H. Hirsh. Feature generation for sequence categorization. In *Proc. 15th Nat. Conf. AI*, pages 733–739, 1998.

[12] N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. *J. Artificial Intelligence*, 76:239–286, 1995.

[13] N. Lesh, N. Martin, and J. Allen. Improving big plans. In *Proc. 15th Nat. Conf. AI*, pages 860–867, 1998.

[14] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[15] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proc. 4th Int. Conf. on KDD*, pages 80–86, 1998.

[16] D.G. Lowe. Similarity metric learning for a variable-kernal classifier. *Neural Computation*, 7(1):72–85, 1995.

[17] D. McDermott. Improving robot plans during execution. In *Proc. 2nd Intl. Conf. AI Planning Systems*, pages 7–12, June 1994.

[18] T. Oates and P. Cohen. Searching for planning operators with context-dependent and probabilistic effects. In *Proc. 13th Nat. Conf. AI*, pages 863–868, 1996.

[19] Richard Segal and Oren Etzioni. Learning decision lists using homogeneous rules. In *Proc. 12th Nat. Conf. AI*, pages 619–25, 1994.

[20] M.J Zaki, N. Lesh, and M. Ogihara. Planmine: Sequence mining for plan failures. In *Proc. 4th Int. Conf. on KDD*, pages 369–373, 1998.