Mining Features for Sequence Classification

Neal Lesh[1], Mohammed J. Zaki[2], Mitsunori Ogihara[3]
lesh@merl.com, zaki@cs.rpi.edu, ogihara@cs.rochester.edu

[1] MERL - Mitsubishi Electric Research Laboratory, 201 Broadway, 8th Floor, Cambridge, MA 02139
[2] Computer Science Dept., Rensselaer Polytechnic Institute, Troy, NY 12180
[3] Computer Science Dept., U. of Rochester, Rochester, NY 14627

Abstract

Classification algorithms are difficult to apply to sequential examples 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 features, which is intractable for large feature sets. We adapt sequence mining techniques to act as a preprocessor to select features for standard classification algorithms such as Naive Bayes and Winnow. Our experiments on three different datasets show that the features produced by our algorithm improve classification accuracy by 10-50%,

1 Introduction

Some classification algorithms work well when there are thousands of features for describing each example (e.g, [Littlestone, 1988]). In some domains, however, the number of potentially useful features is exponential in the size of the examples. Data mining algorithms (e.g., [Zaki, 1998]) 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.

In past work, the rules produced by data mining algorithms have been used to construct classifiers primarily by ordering the rules into decision lists (e.g. [Segal and Etzioni, 1994, Liu *et al.*, 1998]) or by merging them into more general rules that occur in the training data (e.g., [Lee *et al.*, 1998]). In this paper, we convert the patterns discovered by the mining algorithm into a set of boolean features to feed into standard classification algorithms. The classification algorithms, in turn, assign weights to the features which allows evidence from different rules to be combined in order to classify a new example.

While there has been a lot of work on feature selection, it has mainly concentrated on non-sequential domains. In contrast, we focus on sequence data in which each example is represented as a sequence of "events", where each event might be described by a set of predicates. Examples of sequence data include text, DNA sequences, web usage data, and execution traces.

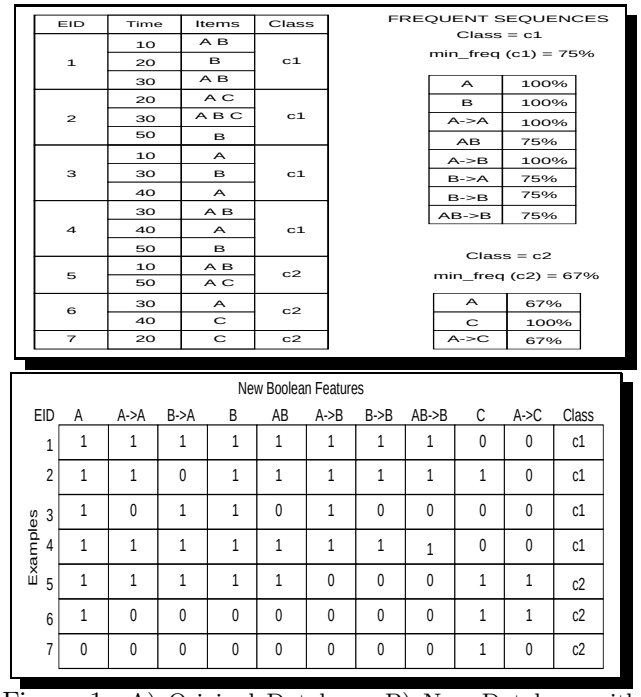
In this paper we combine two powerful mining paradigms: sequence mining, which can efficiently search for patterns that are correlated with the target classes, and classification, which learns to weigh evidence from different features to classify new examples. We present FeatureMine, a scalable disk-based feature mining algorithm. We also specify criteria for selecting good features, and present pruning rules that allow for more efficient feature mining. FeatureMine integrates pruning constraints in the algorithm itself, instead of post-processing, enabling it to efficiently search through large pattern spaces.

2 Data mining for features

We now formulate and present an algorithm for feature mining. Let $\mathcal{F}$ be a set of distinct features, each with some finite set of possible values. Let $\mathcal{I}$ be the set of all possible feature-value pairs. A *sequence* is an ordered list of subsets of $\mathcal{I}$. For example, if $\mathcal{I} = \{A, B, C...\}$, then an example sequence would be $AB \rightarrow A \rightarrow BC$. 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. For example, $AB \rightarrow C$ is a subsequence of $AB \rightarrow A \rightarrow BC$. Let $\mathcal{C}$ be a set of class labels. An *example* is a pair $\langle \alpha, c \rangle$ where $\alpha = \alpha_1 \rightarrow \alpha_2 \rightarrow ... \rightarrow \alpha_n$ is a sequence and $c \in \mathcal{C}$ is a label. Each example has a unique identifier *eid*, and each α_i has a time-stamp at which it occurred. An example $\langle \alpha, c \rangle$ is said to *contain* sequence β if $\beta \prec \alpha$.

Our input database $\mathcal{D}$ consists of a set of examples. This means that the data we look at has multiple sequences, each of which is composed of sets of items. 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, $conf(\beta, c, \mathcal{D}) = fr(\beta, \mathcal{D}_c)/fr(\beta, \mathcal{D})$ where $\mathcal{D}_c$ is the subset of examples in $\mathcal{D}$ with class label c. A sequence is said to be *frequent* if its frequency is more than a user-specified *min_freq* threshold. A rule is said to be *strong* if its confidence is more than a user-specified *min_conf* threshold. Our goal is to mine for frequent and strong patterns. Figure 1 shows a database of examples. There are 7 examples, 4 belonging to class c_1, and 3 belonging to class c_2. In general there can be more than two classes. We are looking for different *min_freq* in each class. For example, while C is frequent for class c_2, it's not frequent for class c_1. The rule $C \Rightarrow c_2$ has confidence $3/4 = 0.75$, while the rule $C \Rightarrow c_1$ has confidence $1/4 = 0.25$.

A *sequence classifier* is a function from sequences to $\mathcal{C}$. A classifier can be evaluated using standard metrics



EID	Time	Items	Class
1	10	A B	c1
	20	B	
	30	A B	
2	20	A C	c1
	30	A B C	
	50	B	
3	10	A	c1
	30	B	
	40	A	
4	30	A B	c1
	40	A	
	50	B	
5	10	A B	c2
	50	A C	
6	30	A	c2
	40	C	
7	20	C	c2

FREQUENT SEQUENCES

Class = c1

min_freq (c1) = 75%

A	100%
B	100%
A->A	100%
AB	75%
A->B	100%
B->A	75%
B->B	75%
AB->B	75%

Class = c2

min_freq (c2) = 67%

A	67%
C	100%
A->C	67%

New Boolean Features

EID	A	A->A	B->A	B	AB	A->B	B->B	AB->B	C	A->C	Class
1	1	1	1	1	1	1	1	1	0	0	c1
2	1	1	0	1	1	1	1	1	1	0	c1
3	1	0	1	1	0	1	0	0	0	0	c1
4	1	1	1	1	1	1	1	1	0	0	c1
5	1	1	1	1	1	0	0	0	1	1	c2
6	1	0	0	0	0	0	0	0	1	1	c2
7	0	0	0	0	0	0	0	0	1	0	c2

Examples

Figure 1: A) Original Database, B) New Database with Boolean Features

such as accuracy and coverage.

Finally, we describe how frequent sequences $\beta_1, ..., \beta_n$ can be used as features for classification. Recall that the input to most standard classifiers is an example represented as vector of feature-value pairs. We represent a example 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 $f_1 = A \rightarrow D$, $f_2 = A \rightarrow BC$, and $f_3 = CD$. The sequence $AB \rightarrow BD \rightarrow BC$ would be represented as $\langle f_1, true\rangle, \langle f_2, true\rangle, \langle f_3, false\rangle$. Note that features can "skip" steps: the feature $A \rightarrow BC$ holds in $AB \rightarrow BD \rightarrow BC$.

2.1 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. However, we do not want to search over the space of all subsets of features [Caruana and Freitag, 1994]), but instead want to evaluate each feature in isolation or by pair-wise comparison to other candidate features. 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. Note that since we use a different *min_freq* for each class, patterns that are rare in the entire database can still be frequent for a specific class. We only ignore those patterns which are rare for any class. 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 together. 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 $M(f_1, \mathcal{D}_{\neg c}) \subseteq M(f_2, \mathcal{D}_{\neg c})$. Intuitively, if f_1 subsumes f_2 for class c then f_1 is superior to f_2 for predicting c because f_1 covers every example of c in the training data that f_2 covers and f_1 covers only a subset of the non-c examples that f_2 covers. The third heuristic leads to two pruning rules, in our feature mining algorithm described below. 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 captures 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 $A \rightsquigarrow B$ 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 $A \rightsquigarrow B$ then any feature containing a set with both A and B will be subsumed by one of its generalizations, and thus we can prune it:

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 $A \rightsquigarrow B$, then α will be subsumed by $\alpha_1 \rightarrow ...\alpha_{i-1} \rightarrow (\alpha_i - B) \rightarrow \alpha_{i+1}... \rightarrow \alpha_n$.*

Feature mining: We can now define the feature mining task. The inputs to the FEATUREMINE algorithm are a set of examples $\mathcal{D}$ and parameters min_freq, 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: Given examples $\mathcal{D}$ and parameters min_freq, 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 $fr(\beta, \mathcal{D}_{c_j}) \geq min\_freq(c_j)$ and $conf(\beta, c_j, \mathcal{D})$ is significantly greater (via chi-squared test) 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 $\mathcal{D}$.

2.2 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. FEATUREMINE is based on the recently proposed SPADE

algorithm [Zaki, 1998] for fast discovery of sequential patterns. SPADE is a scalable and disk-based algorithm that can handle millions of example sequences and thousands of items. Consequently FEATUREMINE shares these properties as well. To construct FEATUREMINE, we adapted the SPADE algorithm to search databases of labeled 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. This enables it to search a large space, where previous methods would fail.

FEATUREMINE 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 algorithm systematically searches the sequence lattice spanned by the subsequence relation, from general to specific sequences, in a depth-first manner.

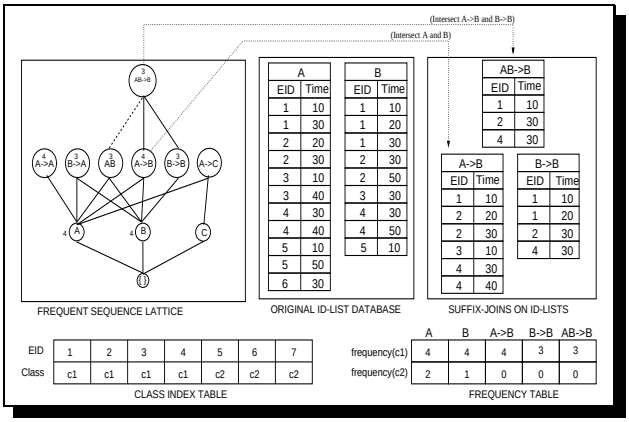


Figure 2: Sequence Lattice and Frequency Computation

Frequency Computation: FEATUREMINE uses a *vertical* database layout, where we associate with each item X in the sequence lattice its *idlist*, denoted $\mathcal{L}(X)$, which is a list of all example IDs (*eid*) and event time (*time*) pairs containing the item. Given the sequence idlists, we can determine the support of any k-sequence by simply intersecting the idlists of any two of its $(k-1)$ length subsequences. A check on the cardinality of the resulting idlist tells us whether the new sequence is frequent or not. Figure 2 shows that the idlist for $A \rightarrow B$ is obtained by intersecting the lists of A and B, i.e., $\mathcal{L}(A \rightarrow B) = \mathcal{L}(A) \cap \mathcal{L}(B)$. Similarly, $\mathcal{L}(AB \rightarrow B) = \mathcal{L}(A \rightarrow B) \cap \mathcal{L}(B \rightarrow B)$. We also maintain the *class index table* indicating the classes for each example. Using this table we are able to determine the frequency of a sequence in all the classes at the same time. For example, A occurs in eids $\{1, 2, 3, 4, 5, 6\}$. However eids $\{1, 2, 3, 4\}$ have label c_1 and $\{5, 6\}$ have label c_2. Thus the frequency of A is 4 for c_1, and 2 for c_2. The class frequencies for each pattern are shown in the *frequency table*.

To use only a limited amount of main-memory FEATUREMINE breaks up the sequence search space into small, independent, manageable chunks which can be processed in memory. This is accomplished via suffix-based partition. We say that two k length sequences are in the same equivalence class or *partition* if they share a common $k-1$ length suffix. The partitions, such as $\{[A], [B], [C]\}$, based on length 1 suffixes are called *parent partitions*. Each parent partition is independent in the sense that it has complete information for generating all frequent sequences that share the same suffix. For example, if a class $[X]$ has the elements $Y \rightarrow X$, and $Z \rightarrow X$. The possible frequent sequences at the next step are $Y \rightarrow Z \rightarrow X$, $Z \rightarrow Y \rightarrow X$, and $(YZ) \rightarrow X$. No other item Q can lead to a frequent sequence with the suffix X, unless (QX) or $Q \rightarrow X$ is also in $[X]$.

FEATUREMINE($\mathcal{D}, min\_freq(c_i)$):
  $\mathcal{P} = \{$ parent partitions, $P_i\}$
  for each parent partition P_i **do** *EnumerateFeatures(P_i)*
ENUMERATEFEATURES(S):
  for all elements $A_i \in S$ **do**
    for all elements $A_j \in S$, with $j > i$ **do**
      $R = A_i \cup A_j$; $\mathcal{L}(R) = \mathcal{L}(A_i) \cap \mathcal{L}(A_j)$;
      if *RulePrune*(R, max_w, max_l) == FALSE **and**
        $frequency(R, c_i) \geq min\_freq(c_i)$ for any c_i
          $T = T \cup \{R\}$; $\mathcal{F} = \mathcal{F} \cup \{R\}$;
    EnumerateFeatures(T);
RULEPRUNE(R, max_w, max_l):
  if *width*(R) $> max_w$ **or** *length*(R) $> max_l$ **return** TRUE;
  if *accuracy*(R) == 100% **return** TRUE;
  return FALSE;

Figure 3: The FEATUREMINE Algorithm

Feature Enumeration: FEATUREMINE processes each parent partition in a depth-first manner, as shown in the pseudo-code of Figure 3. The input to the procedure is a partition, along with the idlist for each of its elements. Frequent sequences are generated by intersecting the idlists of all distinct pairs of sequences in each partition and checking the cardinality of the resulting idlist against *min_sup(c_i)*. The sequences found to be frequent for some class c_i at the current level form partitions for the next level. This process is repeated until we find all frequent sequences.

Integrated Constraints: FEATUREMINE integrates all pruning constraints into the mining algorithm itself, instead of applying pruning as a post-processing step. As we shall show, this allows FEATUREMINE to search very large spaces efficiently, which would have been infeasible otherwise. The *Rule-Prune* procedure eliminates features based on our two pruning rules, and also based on length and width constraints. While the first pruning rule has to be tested each time we extend a sequence with a new item, there exists a very efficient one-time method for applying the $A \rightsquigarrow B$ rule. The idea is to first compute the frequency of all 2 length sequences. Then if $P(B|A) = fr(AB)/fr(A) = 1.0$, then $A \rightsquigarrow B$, and we can remove AB from the suffix partition $[B]$. This guarantees that AB will never appear together in any set of any sequence.

3 Empirical evaluation

We now describe experiments to test whether the features produced by our system improve the performance of the Winnow [Littlestone, 1988] and Naive Bayes [Duda and Hart, 1973] classification algorithms. We ran experiments on three datasets. In each case, we experimented with various settings for min_freq, max_w, and max_l to generate reasonable results. We report the values used, below.

Random parity problems: We first describe a non-sequential problem on which standard classification

algorithms perform very poorly. The problem consists of N parity problems of size M with L distracting, or irrelevant, features. 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 I_k. To generate an instance, we randomly assign each boolean feature true or false with 50/50 probability. An example instance for $N = 3, M = 2$, and $L = 2$ is ($F_{1,1}$=true, $F_{1,2}$=false, $F_{2,1}$=true, $F_{2,2}$=true, $F_{3,1}$=false, $F_{3,2}$=false, I_1=true,I_2= false). There are $N \times M + L$ features, and $2^{N \times M+L}$ distinct instances.

We also choose N weights $w_1, ..., w_N$ which are used to assign each instance one of two class labels (`ON` or `OFF`) as follows. An instance is credited with weight w_i iff the ith set of M features has an even parity. That is, the "score" of an instance is the sum of the weights w_i for which the number of true features in $f_{i,1}, ...f_{i,M}$ 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`. Note that if $M > 1$, then no feature by itself is at all indicative of the class label `ON` or `OFF`, which is why parity problems are so hard for most classifiers. The job of FeatureMine is essentially to figure out which features should be grouped together. Example features produced by FeatureMine are ($f_{1,1}$=true, $f_{1,2}$=true), and ($f_{4,1}$=true, $f_{4,2}$=false). We used a min_freq of .02 to .05, $max_l = 1$ and $max_w = M$.

Forest fire plans: The FeatureMine algorithm was originally motivated by the task of plan monitoring in stochastic domains. As an example domain, we constructed a simple forest-fire domain based loosely on the Phoenix fire simulator [Hart and Cohen, 1992]. 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 before they reach the bases. 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), ..., (time6 Ignite X4 Y8), (time6 Ignite X3 Y8), ..., (time32 Ignite X6 Y1), (time32 Ignite X6 Y0), ...

We form a database of instances from a set of simulations as follows. Because the idea is to predict success or failure before the plan is finished, the instance itself is a list of all events that happen by some time k, which we vary in our experiments. We label each instance with `SUCCESS` if none of the locations with bases have been burned in the final state, or `FAILURE` otherwise. Thus, the job of the classifier is to predict if the bulldozers will prevent the bases from burning, given a partial execution trace of the plan. 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 third row at time 8. Many correlations used by our second pruning rule described in section 2.2 arise in these data sets. For example, $Y8 \rightsquigarrow Ignite$ arises in one of our test plans in which a bulldozer never moves in the eighth column.

Experiment	W	WFM	B	BFM
parity, $N = 5, M = 3, L = 5$	.51	.96	.50	.96
parity, $N = 3, M = 4, L = 8$	.50	.98	.50	1.0
parity, $N = 10, M = 4, L = 10$	.50	.88	.50	.84
fire, time = 5	.60	.79	.69	.81
fire, time = 10	.56	.85	.68	.75
fire, time = 15	.52	.88	.68	.72
spelling, their vs. there	.70	.94	.75	.78
spelling, I vs. me	.86	.94	.66	.90
spelling, than vs. then	.83	.92	.79	.81
spelling, you're vs. your	.77	.86	.77	.86

Table 1: Classification results (W=Winnow, B=Bayes, WFM, BFM = Winnow, Bayes with FeatureMine, resp.)

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: FeatureMine Mining results

For fire data, there are 38 boolean features to describe each *event*. Thus there are $((38 \times 2)^{max_w}))^{max_l}$ possible composite features for describing each sequence of events. In the experiments reported here, we used a $min\_freq = .2$, $max_w = 3$, 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* ([Golding and Roth, 1996]). For each test, we chose two commonly confused words and searched for sentences in the 1-million-word Brown corpus [Kucera and Francis, 1967] 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. These features (for reasons not obvious to us) were significantly correlated with either *there* or *their* in the training set. In the experiments reported here, we used a $min\_freq = .05$, $max_w = 3$, and $max_l = 2$.

3.1 Results

For each test in the parity and fire domains, 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 5,000 examples. Thus, the entire training process required 7,000 examples. We then tested the resulting classifier on 1,000 fresh examples. The results in Tables 1 and 2 are averaged over 25 trials of the process (i.e., we retrained and then re-tested the classifier on fresh examples in each trial). For the spelling correction, we trained on 80 percent of the examples in the Brown corpus and tested on the remaining 20 percent. During training, we mined features from 500 sentences and trained the classifier on all training examples.

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

Experiment	CPU seconds with no pruning	CPU seconds with only $A \rightsquigarrow B$ pruning	CPU seconds with all pruning	Features examined with no pruning	Features examined with only $A \rightsquigarrow B$ 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: results taken from one data set for each example.

tureMine with using only the primitive features themselves, i.e. features of length 1. 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 problems. For the largest parity problem, FeatureMine evaluated more than 7 million features and selected only about 200. There were in fact 100 million possible features (there are 50 booleans features, giving rise to 100 feature-value pairs; we searched to depth $M = 4$.) but most were rejected implicitly by the pruning rules.

Table 3 shows the impact of the $A \rightsquigarrow B$ pruning rule 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 $A \rightsquigarrow B$ pruning, the fire world problems are essentially unsolvable because FeatureMine finds over 20 million frequent sequences.

4 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., [Caruana and Freitag, 1994]). 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.

[Golding and Roth, 1996] 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. [Liu *et al.*, 1998] build decision lists out of patterns found by association mining. [Ali *et al.*, 1997] and [Bayardo, 1997] both combine association rules to form classifiers. Our use of sequence mining is a generalization on association mining. Our pruning rules resemble ones used by [Segal and Etzioni, 1994], 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 algorithm can weigh evidence from many features which each have low accuracy in order to classify new examples.

[Liu and Setiono, 1998] describes recent work on scaling up feature-subset selection. They apply a probabilistic Las Vegas Algorithm to data sets with 16 to 22 features. One of the problems is a parity problem, much like the one described above, which contains 20 features (N=2,M=5,L=10). Their algorithms, thus, search the space of all 2^{20} subsets of the available features. For comparison, we have applied our algorithms to parity problems with 50 features, which results in 100 feature-value pairs. Our algorithm then searches over the set of all conjunctions of up to max_w feature-value pairs. FeatureMine can handle millions of examples and thousands of items, which makes it extremely scalable.

Our work is close in spirit to [Kudenko and Hirsh, 1998], 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 than they have.

References

[Ali *et al.*, 1997] K. Ali, S. Manganaris, and R. Srikant. Partial classification using association rules. In *KDD97*.

[Bayardo, 1997] R.J. Jr. Bayardo. Brute-force mining of high-confidence classification rules. In *KDD97*.

[Caruana and Freitag, 1994] R. Caruana and D. Freitag. Greedy attribute selection. In *ICML94*.

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

[Golding and Roth, 1996] A. Golding and D. Roth. Applying winnow to context-sensitive spelling correction. In *ICML96*.

[Hart and Cohen, 1992] D.M Hart and P.R Cohen. Predicting and explaining success and task duration in the phoenix planner. In *1st Intl. Conf. on AI Planning Systems*.

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

[Kudenko and Hirsh, 1998] D. Kudenko and H. Hirsh. Feature generation for sequence categorization. In *AAAI98*.

[Lee *et al.*, 1998] W. Lee, S. Stolfo, and K. Mok. Mining audit data to build intrusion detection models. In *KDD98*.

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

[Liu and Setiono, 1998] H. Liu and S. Setiono. Some issues on scalable feature selection. In *4th World Congress of Expert Systems: Application of Advanced Info. Technologies*.

[Liu *et al.*, 1998] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *KDD98*.

[Segal and Etzioni, 1994] Richard Segal and Oren Etzioni. Learning decision lists using homogeneous rules. In *AAAI94*.

[Zaki, 1998] M. J. Zaki. Efficient enumeration of frequent sequences. In *7th Intl. Conf. Info. and Knowledge Management*.