Using Conjunction of Attribute Values for Classification [*]

Mukund Deshpande
Dept. of Computer Science & Engineering
Minneapolis, MN 55455
deshpand@cs.umn.edu

George Karypis
Dept. of Computer Science & Engineering
Minneapolis, MN 55455
karypis@cs.umn.edu

ABSTRACT

Advances in the efficient discovery of frequent itemsets have led to the development of a number of schemes that use frequent itemsets to aid developing accurate and efficient classifiers. These approaches use the frequent itemsets to generate a set of *composite features* that expand the dimensionality of the underlying dataset. In this paper, we build upon this work and (i) present a variety of schemes for composite feature selection that achieve a substantial reduction in the number of features without adversely affecting the accuracy gains, and (ii) show (both analytically and experimentally) that the composite features can lead to improved classification models even in the context of support vector machines, in which the dimensionality can automatically be expanded by the use of appropriate kernel functions.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning*

General Terms

Algorithms

Keywords

Classification, SVM, Feature Selection, Conjunctive Attributes, Association Rules

1. INTRODUCTION

After the seminal paper by Agrawal *et al*. [2] on association rules, the field of associating rules and especially its sub-field of frequent itemset generation has seen a great deal of research activity. The extensive research in this field has led to the development of efficient techniques for generating, storing and pruning frequent itemsets [18, 7, 1, 23]. These advances accompanied by growth in the computing power has made the task of frequent itemsets generation much more manageable, than in the past.

As a result, we have witnessed an increased interest in developing schemes that use frequently occurring itemsets to aid in the development of accurate and efficient classification algorithms. To this end, two general approaches have been developed. The first approach uses the frequently occurring itemsets to generate a set of rules, that are then used to build rule-based classifiers [11, 10]. The second approach first expands the dataset's feature space by using the frequently occurring itemsets to form new dimensions, and then uses traditional algorithms to build classification models in that expanded feature space [9, 24]. Despite the differences of these approaches, the common theme that underlies them is that they used the frequently occurring itemsets to generate a set of *composite features*. The idea of using composite features to expand the feature space is not new and has been extensively studied by the machine learning community [17, 26]. Most of these schemes use a greedy approach to find the composite features; hence, they do not search the entire space of all possible attribute-value conjuncts. However, frequent itemset-based approaches have the advantage of exhaustively generating all possible composite features, before selecting which ones to use for classification. Experimental results presented in [11, 10, 24, 9] illustrate that the use of frequently occurring itemsets can lead to measurable improvements in classification accuracy.

In this paper, we build upon this work and further investigate the use of frequently occurring itemsets as composite features for classification. In particular, our research is focused in two directions. First, we investigate the impact of various schemes for selecting the most discriminating set of composite features, and second, we investigate the extent to which the resulting set of composite features can lead to improved classification models in the context of support vector machines, in which the dimensionality can be automatically expanded by the use of appropriate kernel functions. Towards the first direction, we present a variety of schemes that select a set of non-redundant discriminatory composite features, and show that a substantial reduction in the number of features can be obtained, without adversely affecting the accuracy gains achieved by the use of such composite features. Towards the second direction, we show that even though higher order polynomial kernel functions do automatically generate all possible composite features, it is still beneficial to manually expand the feature space by using the discriminatory frequent-itemsets. We prove that a SVM model learnt in the manually expanded feature space will have a lower generalization error than that built by the corresponding higher order polynomial kernel, a fact that was experimentally verified using a set of synthetically generated datasets.

[*]This work was supported by NSF CCR-9972519, 521-6326, EIA-9986042, ACI-9982274, NASA NCC 21231, and Army High Performance Computing Research Center contract number DAAD19-01-2-0014.

The paper is organized as follows, Section 2 presents the related research and Section 3 discusses the terminology used in this paper. Section 4 explains in detail the methodology used for classification, Section 5 presents a detailed analysis of our approach, specifically in context of different classifiers, Section 6 presents the classification results, and finally Section 7 presents the conclusion.

2. RELATED RESEARCH

The idea of using composite features has been well studied in the field of machine learning and goes under the name of *constructive induction*. *Constructive induction* is a process of creating new features/attributes from the task-supplied attributes and then building a model on both these new as well as task supplied attributes [17]. For most cases this approach is diametrically opposite of dimensionality reduction; dimensionality reduction tries to eliminate attributes/features whereas constructive induction expands the feature space before building the classification model. There are many ways of creating new features, Zheng *et al* [26] presents a discussion of using conjunctive, disjunctive and x of N features. The features of type x of N were first studied by Murphy *et al* [15]. Brodley *et al* [3] consider composite features which are modeled as linear functions, which operate on different attribute values. In this paper we will be limiting ourselves to the study of conjunctive attribute-values, a detailed discussion about the advantages of using conjunctive attributes in context of SVM classifier is presented in Section 5.

It is obvious that expanding the feature space to encompass all possible attribute-value conjuncts/disjuncts would make the dataset too large and intractable to build a classification model. Therefore, the main challenge in the field of *constructive induction* is to intelligently search the feature space and select a small set of composite features that leads to an improved classifier, either by improving the accuracy or by improving the understand-ability of the classification model.

Constructive induction has been mainly used in the conjunction with two classification schemes: decision trees and rule based systems. This should not be surprising as using composite features lead to substantially smaller and more understandable decision trees. The methodology used in learning different composite feature based decision trees is roughly the same [6, 12]. First, a decision tree is built on the task-supplied attributes, then a candidate set of composite attributes/features is constructed by taking conjunctions and/or dis-junctions of attributes values along different paths of the decision tree, from this candidate set a small set of composite attributes are retained, which are then incorporated in the dataset and these four steps are repeated in a loop. The process stops when sufficiently accurate decision tree is built. These techniques make use of the attributes selected by the decision tree to restrict the search space of possible composite features.

Composite features have also been used in conjunction with rule based systems. Zheng *et al*, [25] present a modification of the c4.5 rules scheme to use conjunctions/disjunctions of attribute values. In their approach first rules are generated using the traditional c4.5rules [16] scheme, then a candidate set of conjunctive/disjunctive features are generated from these rules, next this candidate set is evaluated to retain a small set of composite attributes.

Liu *et al*, [11] propose a novel technique for using attribute value conjunctions. First, they exhaustively generate all possible attribute-value conjuncts (composite features) using a frequent itemset discovery algorithm, next a pruning scheme is used to eliminate composite features/frequent itemsets that have support and/or confidence below certain threshold. This leaves an extremely small set of composite features. Then considering each composite feature as a rule, a modified version of sequential covering algorithm for rules is run on them to obtain the final ordering of rules. This scheme will be referred as CBA (Class based Associations). Li *et al*. [10] extend the CBA approach by using a modified version of sequential covering algorithm, where an example is eliminated only after it has covered a sufficient number of rules (composite features), this scheme will be referred as CMAR (Classification based on Multiple Association Rules).

3. TERMINOLOGY

The dataset $\mathcal{D}$ used for classification is defined by the tuple $\langle A, C\rangle$, where $A = \{A_1, A_2, A_3, \ldots A_k\}$ are the attributes describing each example in the dataset and C is a finite set of class labels $\{c_1, c_2, c_3, \ldots c_m\}$. Each attribute A_i is assumed to have a finite domain of attribute-values that is know in advance. Note that this model cannot handle continuous attributes and they need to be discretized beforehand [5, 4]. Each example e_i in the dataset $\mathcal{D}$ is represented as $\{(A_1 = a_{1i}, A_2 = a_{2i}, A_3 = a_{3i}, \ldots A_k = a_{ki}), c_i\}$, where a_{1i} corresponds to the attribute-value for attribute A_1 and c_i is the class label assigned to the example e_i. From this user supplied representation a set of composite features will be generated, such that each composite feature A_c represents a conjunction of attribute-values, $A_c = \{(A_1 = a_{1i}) \wedge (A_3 = a_{3j}) \wedge (A_5 = a_{5k})\}$. For example, given the dataset shown in Figure 1, the $\{Outlook = sunny \wedge Windy = True\}$ is an example of a composite feature. Note that a composite feature are formed by taking conjunctions of attribute-value pairs and not just attributes. The *size* of a composite feature is equal to the number of attribute-value pairs present in the composite feature.

An example e_i supports composite feature A_c, represented as $A_c \in e_i$, if all the attribute-value pairs present in A_c are also present e_i. We can define a similar relationship between composite attributes themselves, $A_{c1} \in A_{c2}$, if all the attribute- value pairs present in A_{c1} are also present in A_{c2}. Furthermore, A_{c1} is referred as a *parent* of A_{c2}, and A_{c2} a *child* of A_{c1}, if $size(A_{c2}) = size(A_{c1}) + 1$. Sometimes we will also refer A_{c2} as an extension of A_{c1}. For the example shown in Figure 1 the composite feature $\{Outlook = sunny \wedge Windy = True\}$ is present in examples {1, 4, 5 and 8}. To further simplify this representation we can assign a unique integer to all the unique attribute-value pairs as well as the composite features selected.

4. CLASSIFICATION METHODOLOGY

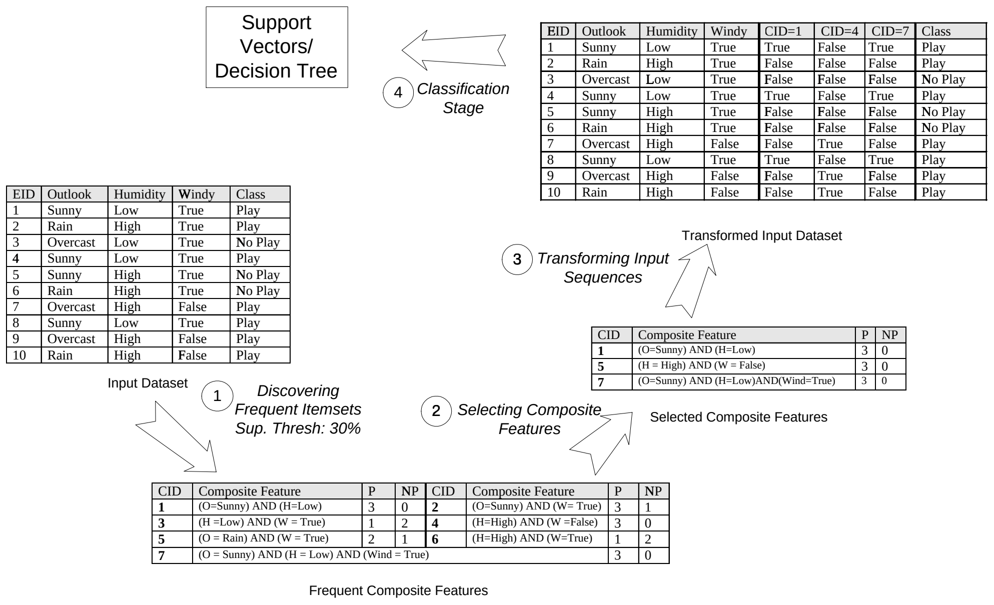
Our methodology for building a classifier using composite features is divided into three subtasks:

1. Generating all the composite features above a support threshold.
2. Pruning this set to obtain a smaller set of composite features.
3. Transforming the user supplied data to incorporate these selected composite features and learning a classifier on this transformed dataset.

These three subtasks are shown in Figure 1.

4.1 Generating Composite Features

This is the first sub-task in our classification procedure; here we generate a set of candidate composite features. For this sub-task we first transform the dataset so that each example is represented as a set of integers, as described in Section 3. We then run a generic frequent itemset discovery algorithm on this dataset assuming each example as a *transaction* and each attribute-value in the example


Support Vectors/ Decision Tree
4 Classification Stage
3 Transforming Input Sequences
Transformed Input Dataset
Input Dataset
1 Discovering Frequent Itemsets Sup. Thresh: 30%
2 Selecting Composite Features
Selected Composite Features
Frequent Composite Features

EID	Outlook	Humidity	Windy	CID=1	CID=4	CID=7	Class
1	Sunny	Low	True	True	False	True	Play
2	Rain	High	True	False	False	False	Play
3	Overcast	Low	True	False	False	False	No Play
4	Sunny	Low	True	True	False	True	Play
5	Sunny	High	True	False	False	False	No Play
6	Rain	High	True	False	False	False	No Play
7	Overcast	High	False	False	True	False	Play
8	Sunny	Low	True	True	False	True	Play
9	Overcast	High	False	False	True	False	Play
10	Rain	High	False	False	True	False	Play

EID	Outlook	Humidity	Windy	Class
1	Sunny	Low	True	Play
2	Rain	High	True	Play
3	Overcast	Low	True	No Play
4	Sunny	Low	True	Play
5	Sunny	High	True	No Play
6	Rain	High	True	No Play
7	Overcast	High	False	Play
8	Sunny	Low	True	Play
9	Overcast	High	False	Play
10	Rain	High	False	Play

CID	Composite Feature	P	NP
1	(O=Sunny) AND (H=Low)	3	0
5	(H = High) AND (W = False)	3	0
7	(O=Sunny) AND (H=Low)AND(Wind=True)	3	0

CID	Composite Feature	P	NP	CID	Composite Feature	P	NP
1	(O=Sunny) AND (H=Low)	3	0	**2**	(O=Sunny) AND (W= True)	3	1
3	(H =Low) AND (W = True)	1	2	**4**	(H=High) AND (W =False)	3	0
5	(O = Rain) AND (W = True)	2	1	**6**	(H=High) AND (W=True)	1	2
7	(O = Sunny) AND (H = Low) AND (Wind = True)					3	0

Figure 1: Various sub-tasks of our classification procedure as defined Section 4

as an *item*. The Frequent Itemset Discovery Algorithm, henceforth referred as FIDA, returns a list of itemsets which occur *frequently* in the dataset. Each itemset represents a composite feature that is a conjunction of all the attribute-values (*items*) making up that itemset. In our procedure we use LPMiner [18] as our FIDA.

The notion of *frequent*, *i.e*, what composite feature is considered as *frequent* is controlled by a user defined parameter to the FIDA called *support threshold*. All the composite features (itemsets) generated by FIDA have a *support* above the *support threshold*. *Support* for a composite feature is defined as the ratio of the number of examples which contain the composite feature to the total number of examples in the dataset. Using a *support threshold*, instead of exhaustively generating composite features, ensures that the discovered composite features are statistically significant. The exact value of *support threshold* is usually dataset dependent and is supplied by the user.

Because the different classes can be of different size, care must be taken to ensure that the composite features properly cover all classes. For this reason we first partition the complete dataset using the class label of the examples into specific *class datasets*. We then run FIDA on each of these *class datasets*. This partitioning ensures that sufficient composite features are discovered for those class labels which occur rarely in the dataset. Next, we combine composite features discovered from each of the *class datasets*. After this step each composite feature has a vector that contains the frequency with which it occurs in each class. Also, to facilitate the efficient execution of the various composite feature selection scheme, which will be described in the next section, we store the composite features into a lattice format. Every composite feature has its child nodes those composite features which can be formed by extending it by one attribute-value pair. The lattice representation makes the task of feature selection extremely efficient.

4.2 Selecting Composite Features

In this sub-task we select a small set of composite features from those generated by the FIDA. There are two motivations behind feature selection: First, the generated composite features contain a lot of noise and redundancy that if eliminated will result in a better classifier. Second, the number of composite features generated by FIDA is quite large and can affect the time needed to build the classification model. Our approach for feature selection is performed in two steps, the first step eliminates redundant composite features and the second step selects the most discriminatory composite features.

4.2.1 Duplicate Elimination

This selection procedure is based on the observation that a large number of the composite features discovered by FIDA are redundant as they provide identical information. As a result, these redundant features can be safely removed without affecting the accuracy of the classifier. Two composite features are said to provide identical information if the set of supporting examples of these two composite features is identical, closed frequent itemset [22].

The lattice representation of the composite features obtained from the FIDA makes the task of identifying duplicates extremely easy. For every node in the lattice we compare its class distribution with its children nodes (single attribute-value extension), and we eliminate the child-node if the frequency distribution is identical. It should be noted that the set of supporting transactions for an extension is a subset of supporting transaction for a composite feature, this is by the subsuming property of the frequent itemsets. Secondly, we always eliminate the longer of the two composite features to ensure that the selected feature generalizes better.

4.2.2 Selecting Discriminatory Composite Feature

A composite feature is considered discriminatory for a particular class if its presence or absence in an example can help in inferring the class label of that example. There are many metrics that evaluate the discriminatory ability of a feature, and in our algorithm we have experimented with two such metrics, *confidence* [2] and *j-measure* [19]. Both of these measures evaluate the discriminatory ability of a composite feature with respect to a particular class label. The confidence of a particular composite feature A_c with respect to class c_i is

$$confidence(A_c, c_i) = \frac{P(A_c, c_i)}{P(A_c)},$$

where $P(A_c, c_i)$ is the probability of observing the composite feature $\boldsymbol{A_c}$ and the class label c_i together in the dataset and $P(A_c)$ is the probability of finding composite feature A_c in the dataset. On the other hand, the *j-measure* of A_c with respect to class c_i is

$$\begin{aligned} \textit{j-measure}(A_c, c_i) &= P(c_i|A_c)\log\left(\frac{P(c_i|A_c)}{P(c_i)}\right) + \\ &\quad (1 - P(c_i|A_c))\log\left(\frac{1 - P(c_i|A_c)}{1 - P(c_i)}\right) \end{aligned}$$

where $P(c_i|A_c)$ is the conditional probability of observing the class c_i given that composite feature A_c is present in the example. Comparing these two metrics we can see that (i) the *confidence* metric takes into account only the presence of a composite feature in an example, whereas the *j-measure* considers both the presence and absence of a composite feature; (ii) both of them can be computed directly from the class distribution of a composite feature.

Both of these metrics compute the discriminatory ability of a composite feature with respect to a class-label and not the composite feature as a whole. The next step is to use these metrics to select a small set of composite features. The procedure used here is similar to the one used for duplicate elimination. We compare each composite feature (A_c) with all of its parents (A_{cp}) using the discriminatory metric, and decide if that composite feature has to be selected or eliminated. There are four possible ways in which this selection can be done depending on the way we compare a composite feature and its parents. Assuming that $D(A_c, c_i)$ is a discriminatory function, either *confidence* or *j-measure,* and A_{cp} is the parent of A_c, we can have:

$$Select(A_c), \quad if\ \forall c_i \quad D(A_c, c_i) > \max_{\forall A_{cp}}(D(A_{cp}, c_i)) \tag{1}$$

$$Select(A_c), \quad if\ \forall c_i \quad D(A_c, c_i) > \min_{\forall A_{cp}}(D(A_{cp}, c_i)) \tag{2}$$

$$Select(A_c), \quad if\ \exists c_i \quad D(A_c, c_i) > \max_{\forall A_{cp}}(D(A_{cp}, c_i)) \tag{3}$$

$$Select(A_c), \quad if\ \exists c_i \quad D(A_c, c_i) > \min_{\forall A_{cp}}(D(A_{cp}, c_i)) \tag{4}$$

Let us consider the right hand side of the equations, if we use the *max* function for selecting the composite feature, it means that the composite feature will be selected only if its discriminatory ability is greater than all of its parents. On the other hand, if the *min* function is used, then a composite feature is selected if it is more discriminatory than at least one of its parents. The *max* function leads to an extremely selective scheme as compared to the *min* function. The left hand side considers the class labels on which the metric can be computed, the condition $\forall\ c_i$ implies that the composite feature has to be more discriminatory w.r.t. all the classes in order for it to be selected; whereas the condition $\exists\ c_i$ implies that the composite feature has to better on any one of the class labels. The condition $\forall c_i$ can never be true if we use the *confidence* metric; because the sum of *confidence* for different class labels is equal to 1.0.

In our scheme we use the Equation 3, *i.e.*, a *max* function and we select a composite feature if it is more discriminatory on any one of the class labels. There are two advantages of this scheme. First, the computation is extremely efficient as it can be done in conjunction with duplicate elimination. Second, we do not need any additional parameters from the user to carry out this scheme, since we always compare a composite feature with its parent.

4.3 Building the classification model

Once we obtain the set of composite features we transform the input dataset into this expanded feature space, each input example is represented as a boolean vector having size equal to the total number of selected composite features. Each element in the vector corresponds to a composite feature and its value is set to true if that composite feature is present in the example, and false otherwise. These boolean vectors are then given to the classifier for building the classification model. We use Support Vector Machines SVM [20] as our classifier; however, the classifier requirements are quite generic and any classifier that can handle boolean vectors as input can be used.

We also use support vector machines to directly classify UCI datasets, *i.e.*, without generating composite features. All continuous attributes in the UCI datasets are first discretized. Then each example is represented as a boolean vector with size equal to the maximum number of attribute-values. As before an element in the boolean vector is set to true, if the input example contains the corresponding attribute- value and false other. After obtaining a boolean vector representation for each example the classification procedure is identical to the one described earlier.

5. ANALYSIS OF PROPOSED APPROACH

In this section we will discuss the advantages of the proposed approach of creating and selecting composite features in the context of Support Vector Machine(SVM) classifiers [20]. Before going into the discussion we describe our terminology and briefly explain the working of the SVM classifier [21].

We assume that we are given l data points $\mathbf{x}_i \in R^n$ labeled $y \in \pm 1$ drawn i.i.d. from a probability distribution $P(\mathbf{x}, y)$. Support vector machines map each example $\mathbf{x} \in R^n$ into a higher dimensional space, possibly infinite, and construct a separating hyperplane in that space, the separating hyperplane acts as a classifier. The mapping of this input space R^n to higher dimensional space $\mathcal{H}$ is represented by $\mathbf{x} \mapsto \Phi(\mathbf{x})$, where different mappings lead to different SVM classifiers. One of the principle advantages of SVM is that even though the learning is done in the higher order space individual examples need not be transformed into this higher order space, and only a kernel function $K(\mathbf{x}, \mathbf{z})$ needs to be defined. The kernel function essentially computes the similarity between $\mathbf{x}$ and $\mathbf{z}$ in higher order space. For example a linear kernel function $K(\mathbf{x}, \mathbf{z})$ is defined as the inner product between two examples $\mathbf{x}$ and $\mathbf{z}$. The classification of an example $\mathbf{x}$ involves computing the distance of the example to the hyperplane in $\mathcal{H}$ and assigning it the class label depending on which side of the hyperplane the example lies. The classification function is represented as,

$$f(x) = w \cdot \Phi(\mathbf{x}) + b = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \tag{5}$$

where, w is the vector defining the hyperplane in the higher dimensional space $\mathcal{H}$, α_i are the weights assigned to the input example x_i that has a class label of y_i. The learning process involves learning the values for α_i and b. The examples that have a non-zero value for α are called support vectors of that model.

Since SVM allows us to operate in higher dimensional spaces one of the first questions to ask is, if it is possible to construct a kernel function that operates in a space represented by the conjuncts of all the attributes. The answer to that question is yes, *polynomial kernel* represented as

$$K(\mathbf{x}, \mathbf{z}) = (< \mathbf{x} \cdot \mathbf{z} > +c)^d$$

operates in a feature space consisting of all possible conjuncts starting from order 1 all the way to order d.

Thus, it would appear that exhaustively generating features outside the classifier is a wasted effort and the same representation can be achieved, potentially in an efficient way, by using a polynomial kernel of suitable degree. However, the key difference between the use of higher-order polynomial kernel functions and our approach is that in addition to finding all frequent itemsets we also perform a feature selection step that eliminates most of the non discriminatory conjuncts. Therefore, the feature space in which our classifier operates, referred as $\mathcal{C}$, is a subset (and generally substantially smaller) than the feature space $\mathcal{H}$ of the polynomial kernel. In light of that, the second question to ask is whether or not there is an advantage in learning a model in $\mathcal{C}$ as opposed to learning a model in $\mathcal{H}$. The answer to this question is yes, and the reason is that even though a model learnt in $\mathcal{C}$, as measured by the value of the classification function $f(x)$ for each example x in the training set, can potentially be learnt in $\mathcal{H}$, the generalization error of $\mathcal{C}$'s model will tend to be lower compared to a model directly learnt in $\mathcal{H}$. These facts are discussed in the rest of this section.

5.1 The equivalence of models learnt in higher order space $\mathcal{H}$ and lower order space $\mathcal{C}$

We first show that the model learnt in the feature space $\mathcal{C}$ is *equivalent* to the model learnt in higher order space $\mathcal{H}$. Two models are said to be *equivalent* if they lead to identical classifications using leave one out estimation scheme, *i.e.*, the output for the classification function defined in Equation 5 is identical in both feature space $\mathcal{C}$ and $\mathcal{H}$. This would imply that the leave one out performance of the classifier is unaffected by operating in lower dimensional space $\mathcal{C}$.

Let $\mathbf{X}_\mathcal{C} = [\mathbf{x}_{\mathcal{C}1}, \mathbf{x}_{\mathcal{C}2}, \ldots, \mathbf{x}_{\mathcal{C}l}]$ be the dataset in space $\mathcal{C}$, and let $\mathbf{X}_\mathcal{H} = [\mathbf{x}_{\mathcal{H}1}, \mathbf{x}_{\mathcal{H}2}, \ldots, \mathbf{x}_{\mathcal{H}l}]$ be the dataset in space $\mathcal{H}$. Since the space $\mathcal{C}$ is a subspace of $\mathcal{H}$, we have

$$\mathbf{X}_\mathcal{H} = \begin{bmatrix} \mathbf{X}_\mathcal{C} \\ \mathbf{X}_\mathcal{P} \end{bmatrix},$$

where $\mathbf{X}_\mathcal{P}$ corresponds to the conjunctive features of all the examples that are pruned. Since we are learning a linear model, the kernel function $K(\mathbf{x}, \mathbf{z})$ is equal to the inner product, $< \mathbf{x} \cdot \mathbf{z} >$, therefore the classification function given in Equation 5 can be represented in matrix notation for an example $\mathbf{x}_i$ as follows,

$$f(\mathbf{x}_i) = \mathbf{D}_\mathcal{C} \mathbf{x}_i^T \mathbf{X}_\mathcal{C} + b_\mathcal{C}, \quad (6)$$

where $\mathbf{D}_\mathcal{C}$ is equal to $[\alpha_{\mathcal{C}1}, \alpha_{\mathcal{C}2} \ldots \alpha_{\mathcal{C}l}] \cdot [y_1, y_2 \ldots y_l]^T$ and corresponds to the model learnt in feature space $\mathcal{C}$. Equation 6 can be represented in matrix notation for all the examples in the dataset $\mathbf{X}_\mathcal{C}$ as follows (assuming leave one out estimation):

$$f(\mathbf{X}_\mathcal{C}) = \mathbf{D}_\mathcal{C} \mathbf{X}_\mathcal{C}^T \mathbf{X}_\mathcal{C} + \mathbf{b}_\mathcal{C}. \quad (7)$$

Similarly for higher dimensional space $\mathcal{H}$ the classification function can be represented as,

$$f(\mathbf{X}_\mathcal{H}) = \mathbf{D}_\mathcal{H} \mathbf{X}_\mathcal{H}^T \mathbf{X}_\mathcal{H} + \mathbf{b}_\mathcal{H}, \quad (8)$$

where $\mathbf{D}_\mathcal{H}$ is the model learnt in the space $\mathcal{H}$. Note that the size of vectors $f(\mathbf{X}_\mathcal{C})$ and $f(\mathbf{X}_\mathcal{H})$ is the same and is equal to the number of examples l. Therefore if the two models are to be *equivalent* (in terms of classification decisions), then $f(\mathbf{X}_\mathcal{H}) = f(\mathbf{X}_\mathcal{C})$, and using Equation 7 and Equation 8 it should be that

$$\begin{aligned} \mathbf{D}_\mathcal{H} \mathbf{X}_\mathcal{H}^T \mathbf{X}_\mathcal{H} + \mathbf{b}_\mathcal{H} &= \mathbf{D}_\mathcal{C} \mathbf{X}_\mathcal{C}^T \mathbf{X}_\mathcal{C} + \mathbf{b}_\mathcal{C} \\ \mathbf{D}_\mathcal{H} &= \mathbf{D}_\mathcal{C} \mathbf{X}_\mathcal{C}^T \mathbf{X}_\mathcal{C} (\mathbf{X}_\mathcal{H}^T \mathbf{X}_\mathcal{H})^{-1} \\ &\quad + (\mathbf{b}_\mathcal{C} - \mathbf{b}_\mathcal{H})(\mathbf{X}_\mathcal{H}^T \mathbf{X}_\mathcal{H})^{-1}. \end{aligned}$$

Thus, we can obtain $\mathbf{D}_\mathcal{H}$, *i.e.*, the weight vector $[\alpha_{\mathcal{H}1}, \alpha_{\mathcal{H}2} \ldots \alpha_{\mathcal{H}l}]$ making up the model, from the model in feature space $\mathcal{C}$ provided that the initial examples are linear independent (*i.e.*, $(\mathbf{X}_\mathcal{H}^T \mathbf{X}_\mathcal{H})^{-1}$ exists).

5.2 Advantage of operating in lower dimensional space $\mathcal{C}$

Even though the hypothesis learnt in the lower dimensional space can be learnt in the higher dimensional space, there is still merit in carrying out feature reduction because of the following reason.

It has been shown that the bounds for error, EP_{err}, in Support Vector machines is given by [21]

$$\text{EP}_{err} \leq \frac{1}{l} E \left\{ \frac{R^2}{M^2} \right\}, \quad (9)$$

where l is the number of examples in the training set, E is the error of the classifier in the training set, R is the radius of the sphere containing all the transformed examples, and M is the maximal margin of separation. Note that both R and M are computed on the transformed space $\mathcal{H}$, for example, in case of a polynomial kernel they will be calculated in the expanded feature space. The key aspect of Equation 9 is that besides the obvious parameters of the training set size and the error on the training set, the generalization ability of the model depends on the ratio R^2/M^2. By operating in a higher-dimensional space, it is possible to learn a model that has a larger margin than a model learnt in a lower dimensional space—leading to a smaller R^2/M^2 ratio and thus better error bounds. On the other hand, as the dimensionality of the space increases, the radius enclosing the instances will also increase, making the error bounds worse. Therefore, choosing the feature space to operate is a trade off between achieving the maximum separating M with the smallest increase in the radius of the space R.

Now consider the model learnt in $\mathcal{C}$ and the corresponding model in $\mathcal{H}$. Let $R_\mathcal{C}$, and $M_\mathcal{C}$, $R_\mathcal{H}$, $M_\mathcal{H}$, be the radii and margins of the two models, respectively. Since the dimensionality of $\mathcal{H}$ is much larger than that of $\mathcal{C}$, $R_\mathcal{H}$ will be much larger than $R_\mathcal{C}$—suggesting that $\mathcal{C}$'s model will have a lower error bound. Of course this can only happen if the relative reduction in the two radii is much greater than the relative reduction of the corresponding maximal margins. However, if the feature selection step ensures that only non-discriminative features are eliminated then the maximal margin $M_\mathcal{C}$ will be similar to $M_\mathcal{H}$. Consequently, the ratio $R_\mathcal{C}^2/M_\mathcal{C}^2$ will be smaller than $R_\mathcal{H}^2/M_\mathcal{H}^2$, leading to a model in $\mathcal{C}$ with lower error bounds. Experiments presented in Section 6 validate this observation.

6. EXPERIMENTAL EVALUATION

We experimentally evaluated the performance of the different composite feature selection techniques on two classes of datasets. The first class contains datasets that were generated synthetically, whereas the second type contains datasets obtained from the UCI dataset repository [13].

Overall, we evaluated four different classification schemes that use different techniques for selecting composite features. These schemes are:

ND-NS This method uses all the composite features that satisfy the minimum support constraint, that is, it does not eliminate redudant composite features and it does not perform any discriminatory feature selection.

IC-NS This method eliminates the composite features that all of its subsets have identical class distribution, but it does not perform any discriminatory feature selection.

IC-Co This method is similar to IC-NS but also eliminates composite features that have a better confidence than all of its parent composite features.

IC-JM This method is similar to IC-Co but instead of confidence, it uses the *j-measure* to determine the discriminating ability of a composite feature.

We performed all the experiments using a 10 way cross validation scheme and computed average accuracy across different runs. We used the SVM implementation provided by the SVMLight [8] package. Our experiments (not reported here) showed that when using a linear kernel function we tend to obtain better results when the examples were normalized to correspond to a unit length vector. However, when the polynomial kernel function was used, keeping the objects unnormalized resulted in better performance. For this reason, in all of our experiments, we use the unit-length normalization for linear kernel functions, and leave the objects un-normalized for polynomial kernel functions.

6.1 Evaluation on Synthetic Datasets

To experimentally validate the analysis presented in Section 5, we created a series of synthetic datasets, designed so that composite features are essential for the proper classification. This allow us to evaluate the impact of using all composite features or the features selected using the various composite feature selection schemes presented in this paper.

6.1.1 Synthetic Dataset Characteristics

To keep our analysis simple, the synthetic dataset contains just two classes. The attributes making up the dataset are of two kinds, *relevant* and *irrelevant*. The relevant attributes influence the class label, whereas irrelevant do not, and can be thought of as noise. For our experiments we restricted the number of relevant attributes to three and varied the number of irrelevant attributes. The cardinality (number of attribute-values) of the relevant and irrelevant attributes is the same, equal to four and each of the attribute-value is equally likely, *i.e.*, uniform distribution of attribute-values for all the attributes.

The classification function, which determines the class label of an example, was constructed so that the class label depends only on the conjuncts of the relevant attribute-values. Since there are three relevant attributes each with four possible attribute-values, we can have 64 possible conjuncts. These 64 conjuncts are equally divided into two sets with each set corresponding to one particular class label. Hence both the classes are equally likely and are determined by the set in which the conjunct of the relevant attributes belongs. Furthermore, we have taken care to ensure that the class distribution with respect to single relevant attribute-value is uniform. In other words the conditional probability of class label given the value of single relevant attribute is 0.5, whereas the conditional probability of a class label given all three relevant attribute-values is 1.0.

The number of non-relevant attributes control the dimensionality of the problem. To make the dataset realistic we also added some noise to the classification function. The amount of noise is controlled by a parameter called *R value*, *R* refers to the randomness. While assigning the class label to an example for most of the time we use the classification function, however once in a while we invert the result of the classification function. Specifically, if we set the *R value* to be 0.95, then for 95% of times we will assign the class label according to the classification function and 5% of times we will invert the class label.

This framework allow us to generate different datasets by varying the number of irrelevant attributes and the *R* value. For our experiments we generated 24 datasets, by using {0, 5, 10, 15, 20, 25} as the possible values for the number of irrelevant attributes, and {1.0, 0.9, 0.8, 0.7} as the possible values for *R*. For each of these paramater combinations we generated a dataset containing 5000 examples.

6.1.2 Evaluating Performance on Synthetic Datasets

Using the 24 datasets described in the previous section, we evaluated the classification performance of five different schemes. Four of these schemes use the linear SVM kernel function on the feature spaces produced by the four composite feature selection schemes (*ND-NS, IC-NC, IC-CO, IC-JM*), and one scheme corresponds to a third-order polynomial SVM kernel operating in the original feature space. Note that the order of the polynomial kernel was selected so that to match the number of composite features that are required to properly classify these datasets. For all the composite-feature based classification schemes the maximum size of the composite feature is restricted to 3 and the support threshold for FIDA is set to 1%. Since the size of each dataset is 5000, and each class-defining composite feature occurs on the average $5000/4^3 \approx 78$ times, a 1% support ensures that almost all class-defining composite features are being discovered.

The accuracy of a classifier on the test set depends on two main factors: first, the quality of the model *i.e.*, the ability of the model to capture the hypothesis in the training set. Second, the generalizability of the model on the testset (unseen examples). In an attempt to decouple these two effects, our evaluation will be done in two steps. First, we will compare the various models on how effective they are in classifying the *training* set. The resulting accuracy, sometimes called the *retrained accuracy* [14] of the model, can be used to evaluate the extent to which a classify is able to learn a reasonably good model for the training set. Second, we will compare how well these models generalize to unseen cases, by computing the accuracy they obtain on the *test* set.

6.1.2.1 Training Set Performance.

Table 1 displays the accuarcy of the different classifiers on the training set. The accuracies of SVM with polynomial kernel *Poly*, and the two composite feature selection schemes *ND-NS* and *IC-NC*, are within 1% of each other. However, the accuracy for *IC-CO* and *IC-JM* are in general better (and in some cases by a significant amount) than the accuracies of the other schemes. This result may be surprising at first, because the feature space that *IC-CO* and *IC-JM* operate is in general a subset of the feature spaces of the other schemes, and as the analysis in Section 5.1 showed, the model learnt in the space with fewer features can also be learnt in a *superset* feature space. However, the SVM formulation that was used to learn these models employs soft margins, which trade miss-classification cost for a larger margin. As a result, it is possible that the different models are optimized at different accuracies, since the spaces that they operate are different. Also note that for some values of *R*, as the number of irrelevant features increases the accuracy initially decreases, but after some point it actually gets better. This is because as more features become available, SVM is able to find a better model, using such irrelevant features. However, that also represents the point at which SVM is overfitting the dataset, and these training set improvements do not lead to improvements in the test set.

Table 1: Accuracy on the training set.

# irr.	*R = 1.0*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	100.00	100.00	100.00	100.00	100.00
5	96.01	99.02	99.02	99.98	100.00
10	86.04	89.23	89.23	92.58	97.79
15	88.74	90.77	90.77	93.80	96.21
20	92.52	93.90	93.90	96.44	97.95
25	94.93	95.92	95.92	98.14	99.06
# irr.	*R = 0.9*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	89.82	89.82	89.82	89.82	89.82
5	85.90	88.91	88.91	90.57	90.32
10	83.68	86.45	86.45	89.36	92.63
15	87.60	89.75	89.75	92.60	94.35
20	92.14	93.51	93.51	96.21	97.45
25	94.02	95.06	95.06	97.83	98.95
# irr.	*R = 0.8*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	79.58	79.58	79.58	79.58	79.58
5	77.69	80.29	80.29	82.08	80.89
10	81.66	84.29	84.29	86.62	88.29
15	86.83	88.90	88.90	91.70	93.06
20	91.19	92.62	92.62	95.65	96.96
25	94.34	95.44	95.44	97.99	98.92
# irr.	*R = 0.7*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	69.80	69.78	69.78	69.78	69.78
5	73.03	75.17	75.17	76.26	76.36
10	80.61	83.23	83.23	85.46	85.88
15	86.73	88.69	88.69	91.35	92.30
20	90.82	92.27	92.27	95.47	96.52
25	94.15	95.18	95.18	97.79	98.76

6.1.2.2 *Test Set Performance.*

Table 2 shows the accuracy of the various classifiers on the test set.

Looking at these results we can see that SVM with polynomial kernel (*Poly.*) starts off with an accuracy of 100% for $R = 1.0$ and 0 irrelevant attributes, but the accuracy drops sharply to 59% with 5 irrelevant attributes, and the accuracy further drops to 50% when the number of irrelevant attributes is increased to 10. On the other hand composite features based schemes, especially *IC-CO*, show considerable resilience against the number of irrelevant attributes, with the accuracy of *IC-CO* degrading to only 80% when the number of irrelevant attributes is increased to 25. Similarly, composite features based schemes are also more resilient to noise than SVM with the polynomial kernel. Next, even the most primitive of the composite feature selection based scheme *ND-NS* that only uses support threshold for feature selection, also consistently outperforms the SVM classifier with polynomial kernel for all possible settings; further validating the importance of feature selection for improving the accuracy of the classifier. Among the two feature selection schemes *IC-CO* & *IC-JM*, *IC-CO* consistency outperforms the *IC-JM*. *IC-CO* also shows remarkable resilience to the number of irrelevant attributes. The reason that *IC-CO* outperforms *IC-JM* is because the feature selection strategy based on *j-measure* is extremely aggressive and results in the pruning of useful features.

6.1.2.3 *Generalizability Comparison.*

To compare the generalizability of the composite feature based schemes over that of the SVM classifier using polynomial kernel(*Poly*) we compute the ratio of accuracy of a composite feature based scheme and the *Poly* scheme. We do this for each of the composite feature based schemes, on all 24 datasets, and for both the accuracy on the training- as well as test-set. The ratio of training set accuracies give us the improvement in the quality of the model for a composite feature based scheme versus the *Poly* scheme; this ratio is referred as *train ratio*. The ratio of test set accuracies reflects the improvement in both the quality as well the generalizability of the model; it is referred as the *test ratio*. Finally, we take the ratio of the *test ratio* and the *train ratio*, this ratio is called the *generalizability improvement*. If this ratio is greater that 1 then it indicates that the generalizability of the composite feature scheme is better than the generalizability of the *Poly* scheme.

Table 2: Accuracy on synthetic datasets.

# irr.	*R = 1.0*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	100.00	100.00	100.00	100.00	100.00
5	79.100	96.00	95.70	100.00	100.00
10	51.080	61.00	61.34	100.00	81.04
15	51.400	55.00	55.26	100.00	61.92
20	50.700	53.00	53.28	94.22	58.10
25	50.120	50.00	50.16	80.90	51.98
# irr.	*R = 0.9*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	89.82	90.00	89.82	89.82	89.82
5	65.94	79.00	79.24	90.32	90.32
10	52.44	58.00	57.90	90.78	70.82
15	50.82	51.00	52.54	88.88	56.22
20	50.80	54.00	53.04	72.26	56.04
25	50.80	51.00	50.50	65.94	52.58
# irr.	*R = 0.8*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	79.58	79.54	79.58	79.58	79.58
5	55.70	65.00	65.44	79.76	78.84
10	50.82	55.00	54.38	80.70	60.54
15	49.62	50.00	51.20	67.94	51.72
20	50.62	53.00	52.76	60.74	53.84
25	50.78	49.00	48.70	58.24	50.30
# irr.	*R = 0.7*				
Attr.	*Poly.*	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	69.52	68.00	69.58	67.22	69.50
5	53.68	58.00	57.16	65.06	61.92
10	50.56	52.00	51.90	62.02	54.84
15	50.34	50.00	50.42	56.08	51.66
20	50.84	52.00	51.60	53.82	52.34
25	50.70	49.00	48.56	52.54	49.98

Table 3 displays the *generalizability improvement* for four composite feature based schemes on all the different datasets. For most all the dataset the generalizability improvement has a value greater than 1 or very close to 1; this implies that feature selection does lead to better generalizability. Of the four schemes we find that the greatest generalizability improvement is seen for the *IC-CO* scheme, with the generalizability improvement value as high 1.841, indicating an improvement of about 84%. We also notice that the value of generalizability improvement varies with the number of irrelevant attributes and peaks at one particular value, and then decreasing as the number of irrelevant attributes are increased. This indicates the generalizability of the composite feature based schemes is also affected as the number of irrelevant attributes is increased and the scheme performs optimaly for one value or number irrelevant attributes.

6.2 Evaluation on Real Datasets

Our overall methodology for building classifiers using composite features, and the various pruning schemes, were also evaluated using a set of problems from the UCI dataset collection [13]. This particular set of problems was selected because they have been used in many previous classification algorithm studies [11, 10], even though they may not be well-suited for the classification models that can be learnt by our algorithm. The dataset characterisitics of

Table 4: Overall comparison of accuracy.

Dataset	Dataset Characteristics			*CBA*	*CMAR*	*C4.5*	*SVM*				
	# Attr.	# Cl.	# Ex.				*Direct*	*ND-NS*	*IC-NS*	*IC-CO*	*IC-JM*
anneal	38	6	898	97.90	97.30	94.76	98.33	98.44	97.89	98.33	99.00
austra	14	2	690	84.90	86.10	85.22	85.80	86.38	86.96	86.09	85.80
breast	10	2	699	96.30	96.40	95.42	97.14	97.00	96.86	96.71	97.29
cleve	13	2	303	82.80	82.20	80.17	84.11	84.44	83.47	83.46	83.45
crx	15	2	690	84.70	84.90	84.93	86.09	86.67	86.52	86.09	84.50
diabetes	8	2	768	74.50	75.80	76.18	77.35	78.14	78.53	78.53	77.88
german	20	2	1000	73.40	74.90	72.70	75.70	71.90	72.30	72.60	74.30
glass	9	7	214	73.90	70.10	65.97	75.33	78.61	78.12	75.28	73.46
heart	13	2	270	81.90	82.20	80.00	82.96	85.56	85.56	84.82	82.96
hepati	19	2	155	81.80	80.50	83.25	84.38	79.29	81.21	81.21	85.79
horse	22	2	368	82.10	82.60	82.92	85.61	82.61	84.79	83.43	82.32
iris	4	3	150	94.70	94.00	95.33	93.33	94.00	94.00	94.00	93.33
labor	1	2	57	86.30	89.70	79.00	89.33	77.33	94.67	94.67	94.67
led7	7	10	3200	71.90	72.50	72.88	72.41	71.25	72.19	73.03	72.78
lymph	18	4	148	77.80	83.10	79.72	84.38	81.05	80.33	81.67	80.38
pima	8	2	768	72.90	75.10	74.22	77.34	78.78	79.04	78.52	78.52
tic-tac	9	2	958	99.60	99.20	98.64	95.41	98.54	98.54	96.97	97.70
wine	13	3	178	95.00	95.00	92.75	99.44	98.86	98.30	99.44	98.86
zoo	16	7	101	96.80	97.10	92.09	96.00	97.00	92.09	96.00	96.00
Average				83.90	84.38	82.95	85.61	85.57	85.68	85.67	85.58

Table 3: Ratio of the improvement of composite feature based scheme versus *Poly* scheme on test set and the training set.

# irr.	*R = 1.0*			
Attr.	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	1.00	1.00	1.00	1.00
5	1.17	1.17	1.21	1.21
10	1.15	1.15	1.81	1.39
15	1.04	1.05	1.84	1.11
20	1.03	1.03	1.78	1.08
25	0.98	0.99	1.56	0.99
# irr.	*R = 0.9*			
Attr.	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	1.00	1.00	1.00	1.00
5	1.15	1.16	1.29	1.30
10	1.07	1.06	1.62	1.22
15	0.97	1.00	1.65	1.02
20	1.04	1.02	1.36	1.04
25	0.99	0.98	1.24	0.98
# irr.	*R = 0.8*			
Attr.	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	0.99	1.00	1.00	1.00
5	1.12	1.13	1.35	1.36
10	1.04	1.03	1.49	1.10
15	0.98	1.00	1.29	0.97
20	1.03	1.02	1.14	1.00
25	0.95	0.94	1.10	0.94
# irr.	*R = 0.7*			
Attr.	*ND-NS*	*IC-NC*	*IC-CO*	*IC-JM*
0	0.97	1.00	0.96	1.00
5	1.05	1.03	1.16	1.10
10	0.99	0.99	1.15	1.01
15	0.97	0.97	1.05	0.96
20	1.00	0.99	1.00	0.96
25	0.95	0.94	0.99	0.94

these problems are displayed in first three columns in Table 4.

In this set of experiments, besides the four algorithms based on composite features, we also used four additional classifiers. These were C4.5 [16], CBA [11], CMAR [10], and SVM operating solely on the original features of each dataset (after being descritized using the method described in Section 4.3). To differentiate this SVM-based classifier from the ones using composite features, we will refer to it as *direct* SVM. The CBA and CMAR classifiers are two other schemes based on composite features that were previously developed and were briefly described in Section 2. For the composite feature based classifiers we used a support threshold of 1.0% for all the datasets, except `hepati, horse` where we used a support threshold of 2.0% and for `lymph and zoo` where we used the support threshold of 5.0%. This was done to ensure that the composite features generated are statistically significant. For SVM classification we used radial basis function kernel.

Table 4 displays the accuracy values obtained by the different classification algorithms for each dataset. The last row in this table labeled "Average" shows the accuracy achieved by each classifier averaged over the different datasets. From these results we can see that the SVM-based schemes (with and without composite features) achieve higher accuracies than that obtained by either C4.5, CBA, or CMAR. However, between the different SVM-based schemes we can see that on the average, there is little difference between the schemes that use composite features and direct SVM, suggesting that composite features are not beneficial for these datasets.

7. CONCLUSION

In this paper we presented a number of classification algorithms that use frequent itemsets to expand the feature space and evaluated a variety of schemes for selecting discriminating composite features. Our experimental results show that the proposed schemes can substantially reduce the number of composite features used, which improves the classification accuracy. Moreover, we have both analytically and experimentally shown that the pruned composite feature space reduces the generalization error obtained by support vector machines, leading to better classifiers.

8. REFERENCES

[1] R. Agrawal, C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. In *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, August 2000.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. of 1993 ACM-SIGMOD Int. Conf. on Management of Data*, Washington, D.C., 1993.

[3] C. E. Brodley and P. E. Utgoff. Multivariate versus univariate dcision trees. Technical report, University of Massachusetts, 1992.

[4] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretisation of continuous features. In *Machine Learning: Proceedings of the Twelfth Internation Conference*, 1995.

[5] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

[6] P. G. and H. D. Boolean feature discovery in empirical learning. *Machine Learning*, 1990.

[7] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX*, May 2000.

[8] T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press, 1999.

[9] N. Lesh, M. J. Zaki, and M. Ogihara. Mining features for sequence classification. In *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1999.

[10] W. Li, J. Han, and J. Pei. Cmar: Accurate and efficient classification based on multiple class-association rules. In *IEEE International Conference on Data Mining*, 2001. Also available as a UMN-CS technical report, TR# 01-026.

[11] B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *4th Internation Conference on Knowledge Discovery and Data Mining*, 1998.

[12] C. J. Matheus and L. Rendell. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artifical Intelligence*, 1989.

[13] C. Merz and P. Murphy. UCI repository of machine learning databases, 1998.

[14] T. M. Mitchell. *Machine Learning*. Mc Graw Hill, 1997.

[15] P. M. Murphy and M. J. Pazzani. Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. In *Proc. of the 8th IntẆorkshop on Machine Learning*, 1991.

[16] J. R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[17] M. R. S. Pattern recognition as knowledge guided computer induction. Technical report, University of Illinois at Urbana Champaign, 1978.

[18] M. Seno and G. Karypis. Lpminer: An algorithm for finding frequent itemsets using length-decreasing support constraint. In *IEEE International Conference on Data Mining*, 2001. Also available as a UMN-CS technical report, TR# 01-026.

[19] P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE transactions on Knowledge and Data Engineering*, 4(4):301–316, August 1992.

[20] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[21] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection fof svms. *Advances in Neural Information Processing Systems*, 2000.

[22] M. J. Zaki. Generating non-redundant association rules. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 34–43. ACM Press, 2000.

[23] M. J. Zaki. Scalable algorithms for association mining. *Knowledge and Data Engineering*, 12(2):372–390, 2000.

[24] M. J. Zaki, N. Lesh, and O. Mitsunari. Planmine: Predicting plan failures using sequence mining. *Intelligence Review, special issue on the Application of Data Mining*, 2000.

[25] Z. Zheng. Constructing conjunctive attributes using production rules. *Journal of Research and Practice in Information Technology*, 2000.

[26] Z. Zijian. A comparison of constructive induction with different types of new attribute. Technical report, School of Computing and Mathematics, Deakin University, Geelong, Victoria, Australia, 1996.