Using Conjunction of Attribute Values for Classification*

Mukund Deshpande and George Karypis

University of Minnesota, Department of Computer Science/Army HPC Research Center
Minneapolis, MN 55455

Technical Report #02-011

Abstract

Advances in the efficient discovery of frequent itemsets in large databases have led to the development of a number of schemes that use frequent itemsets to aid in the development of 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 feature space can lead to improved classification models in the context of support vector machines, in which the dimensionality can automatically be expanded by the use of appropriate kernel functions.

Keywords: Classification, SVM, Feature Selection, Conjunctive Attributes

1 Introduction

After the seminal paper by Agrawal *et al*. [AIS93] 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 [SK01, HPY00, AAP00, Zak00]. 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 [LHM98, LHP01]. The second approach first expands the dataset's feature space by using the frequently occurring itemsets, and then uses traditional algorithms to build classification models in that expanded feature space [LZO99, ZLM00]. 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

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

of using composite features to expand the feature space is not new and has been extensively studied by the machine learning community [S.78, Zij96]. 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 [LHM98, LHP01, ZLM00, LZO99] 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.

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 [S.78]. 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* [Zij96] 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* [MP91]. Brodley *et al* [BU92] 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 different classifiers 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 model.

Constructive induction has been mainly used in the conjunction with two classification schemes: decision trees and rule based systems, with majority of the work done on decision trees. This should not be surprising as using composite features lead to substantially smaller and more understandable decision trees. The methodology used for different decision tree learning is broadly the same [GD90, MR89]. First, a decision tree is built on the task-supplied attributes, then a candidate set of composite attributes 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.

Composite features have also been used in conjunction with rule based systems. Zheng *et al*, [Zhe00] 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 [Qui93] 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.

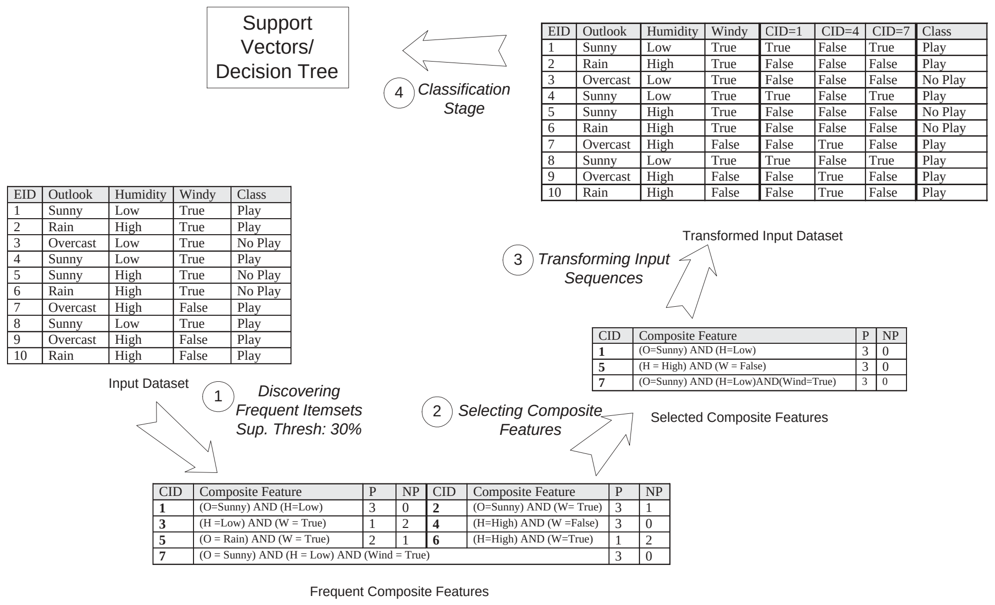
Liu *et al*, [LHM98] propose a novel technique for using attribute value conjunctions. First, they exhaustively generate all possible attribute value conjuncts (composite features) using a frequent itemset discovery algorithm, next a pruning scheme is used to eliminate composite features/frequent itemsets 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.* [LHP01] 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 $< A, C >$, 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 [FI93, DKS95]. 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 feature A_c represents a conjunction of attribute values, $A_c = \{(A_1 = a_1 i) \wedge (A_3 = a_{3j}) \wedge (A_5 = a_{5k})\}$. 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 an *extension* of A_{c2}, if $size(A_{c2}) = size(A_{c1}) + 1$. For the example shown in Figure 1 the composite feature $\{Outlook = sunny \wedge Windy = True\}$ is present in example (1, 4, 5 and 8).

After selecting all the composite features, each example is transformed so that each example, in addition to the user supplied attributes, it also contains the composite features it supports. To further simplify this representation we can assign a unique integer to all the unique attribute value pairs as well as the composite attribute selected.


Support Vectors/ Decision Tree
4 Classification Stage
Transformed Input Dataset
3 Transforming Input Sequences
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

4 Classification Methodology

In this section we describe in detail our methodology for building a classifier using composite features. This task is divided into three subtasks:

1. Generating all the composite features above a support threshold
2. Pruning this set of 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 steps 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 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 [SK01] 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 and not random noise. 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 in our algorithm, 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 of the dataset 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 dataset*. 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 feature into a lattice format. Every composite feature has its child nodes those composite feature 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 can be easily eliminated, and will result in a better classifier. Second, the number of composite features generated by FIDA is quite large and will 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 whereas the second step selects the most discriminatory composite features.

4.2.1 Duplicate Elimination

This selection procedure is based on the observation that the composite features discovered by FIDA contain a lot of redundancy *i.e*., they provide identical information. As a result these duplicate 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.

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 child 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

In this step we further eliminate features obtained from the previous step to retain only those composite features which are considered discriminatory. 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 which evaluate the discriminatory ability of a feature, and in our algorithm we have experimented with two such metrics, *confidence* [AIS93] and *j-measure* [SG92]. Both these measures evaluate the discriminatory ability of a composite feature w.r.t. a particular class label. The confidence of a particular A_c w.r.t. class c_i is defined as

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

, where $P(A_c, c_i)$ is the probability of observing the composite feature 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 w.r.t. class c_i is defined as follows:

$$j - measure(A_c, c_i) = P(c_i|A_c).\log\left(\frac{P(c_i|A_c)}{P(c_i)}\right) + (1 - P(c_i|A_c)).\log\left(\frac{1 - P(c_i|A_c)}{1 - P(c_i)}\right),$$

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. Studying the two formulae we can make two observations: First, *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. Second, both of these metrics can be computed directly from the class distribution of a composite feature. Also note that all 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. Assuming that $D(A_c, c_i)$ is a discriminatory function and A_{cp} is the parent of A_c, we can have:

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

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

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

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

Lets 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 max function and select a composite feature if it is more discriminatory on any one of the class labels, as mentioned before we use both *confidence* and *j-measure* as our metric. There are many 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 parameter from the user to carry out this scheme.

4.3 Building the classification model

Once we obtain the set of composite features we transform the input dataset into this expanded feature space. Now each input example is represented as a boolean vector of 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 given to the classifier for building the classification model. We have experimented with two classification schemes, C4.5 [Qui93] and SVM [Vap98]. We will next discuss some implementation specific details about the classifiers.

We use a locally modified version of C4.5, known as C4.5-sparse, to handle the sparse and high dimensional data generated as a result of transforming the datasets. C4.5-sparse stores the data in a sparse format and hence can easily handle high dimensional datasets. Support vector machines operate only on continuous attributes, hence we first normalize each boolean vector to be of unit length and then feed this normalized vector to the SVM Classifier. We also use SVM to directly classify the UCI datasets, in this case we first discretized all the continuous attributes. Then, we transformed this discretized representation into a feature space such that each attribute value corresponds a new attribute. This allows us to again represent the examples as a boolean vector.

5 Analysis of Proposed Approach

In this section we study the advantages of using composite features in the context of two classifier: decision trees and SVM. Since the classification methodology of these two classifiers differs a lot, we consider each one of them separately.

5.1 Decision Trees

Zheng *et al* [Zhe00] provide a detailed discussion of advantages of using conjunctive attributes in context of decision trees. *Composite Features* provide a way of overcoming two of the well known short comings of C4.5 [Qui93], namely *fragmentation* and *replication*. Before we go on to discuss them, it should be noted that composite features do not provide any additional expressive power to C4.5. The classification function learnt by using composite feature can also be learnt by directly using C4.5, provided we have a well spread out data and there is no noise in it. However both of these two conditions are almost never fulfilled in practice.

We will study the *fragmentation* problem first; the decision tree is constructed by recursively partitioning the attribute space using one attribute at at time, each of these partitions is represented by a node in the decision tree. Figure 2(a) displays an example dataset and the corresponding decision tree. As can be seen from the figure the decision tree needs to partition the dataset four times to capture all the examples. This repeated partitioning reduces its generalizability and can result in lower accuracy of the classifier on the test set. On the contrary composite features allow us to succinctly capture the concept present in the dataset and as a result the decision tree is very compact.

Another problem observed in decision trees is *replication*; in replication a portion of a subtree is constructed multiple times. An example of *replication* is shown in Figure 3(a). *Replication* also leads to decision trees which are deep and difficult to understand. *Composite features* in some cases can eliminate replication. One such example where composite features help is shown in Figure 3(b).

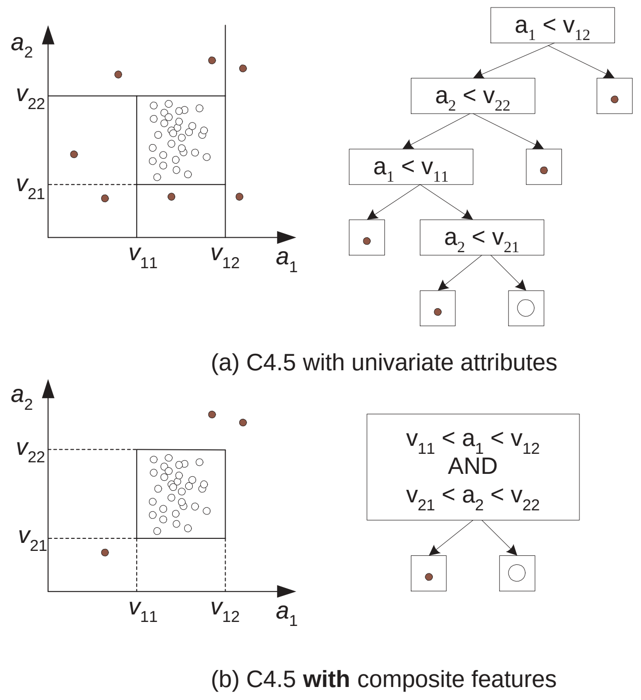

a_2
v_{22}
v_{21}
v_{11}
v_{12}
a_1
$a_1 < v_{12}$
$a_2 < v_{22}$
$a_1 < v_{11}$
$a_2 < v_{21}$
(a) C4.5 with univariate attributes
$v_{11} < a_1 < v_{12}$ AND $v_{21} < a_2 < v_{22}$
(b) C4.5 **with** composite features

Figure 2: Fragmentation due to C4.5, (a) displays decision tree when composite features are not used (b) decision tree with composite features.

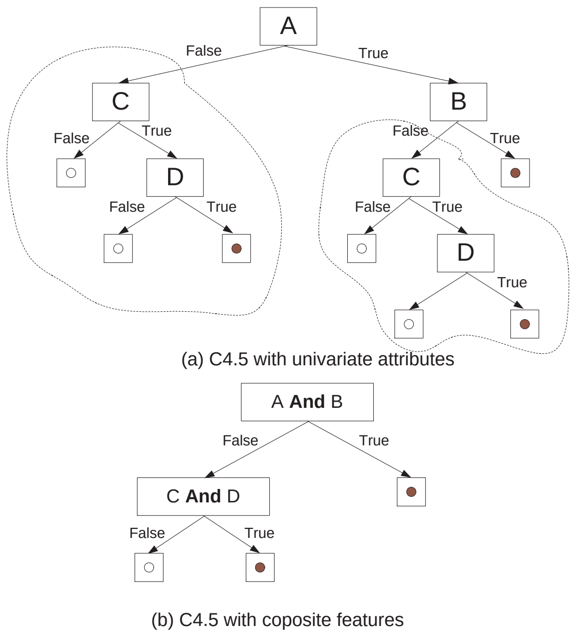

A
False
True
C
B
False
True
D
(a) C4.5 with univariate attributes
A **And** B
C **And** D
(b) C4.5 with coposite features

Figure 3: Replication in C4.5 (a) displays a decision tree in which a sub tree is replicated (b) decision tree using composite features.

5.2 SVM

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 [Vap98]. Before going into the discussion we describe terminology and briefly explain the working of the SVM classifier [WMC$^+$00].

We assume that we are given l data points $\mathbf{x}_i \in R^n$ labeled $y \in \pm y$ drawn i.i.d. from a probability distribution $P(\mathbf{x}, y)$. Support vector machines can map each example $\mathbf{x} \in R^n$ into a higher dimensional space, possibly infinite, and construct a separating hyperplane in that space. 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. For a simple case the kernel function defines the inner product between two examples in the expanded feature space *i.e.*, similarity 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 in the class label depending on which side of the hyperplane the example lies. The classification function is represented as,

$$f(x) = w \cdot \Phi(\mathbf{x}) + b = \sum_i \alpha_i^0 y_i K(\mathbf{x}_i, \mathbf{x}) + b,$$

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 which has a class label of y_i. The learning process involves learning the values for α_i and b. The examples which 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 which will operate 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. The polynomial kernel operates on $\binom{n+d-1}{d}$ distinct features, which is essentially all possible conjuncts starting from order 1 to order d.

From this discussion 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 key 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 proven in the rest of this section.

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 subspace of $\mathcal{H}$, we have

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

where $\mathbf{X}_{\mathcal{P}}$ are the conjunctive features from all the examples which 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 5.2 can be represented in matrix notation for an example $\mathbf{x}_i$ as

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

where $\mathbf{D}_{\mathcal{C}}$ is equal to $[\alpha_{\mathcal{C}1}, \alpha_{\mathcal{C}2} \ldots \alpha_{\mathcal{C}l}] \cdot [y_1, y_2 \ldots y_l]^T$. Similarly, Equation 5.2 can be represented in matrix notation for all the examples in the dataset $\mathbf{X}_{\mathcal{C}}$ as follows (assuming leave one out classification):

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

The classification of l examples by learning a linear model in higher dimensional space $\mathcal{H}$ can be represented as

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

where $\mathbf{D}_{\mathcal{H}}$ is the model learnt in the space $\mathcal{H}$. Note that the dimension of $f(\mathbf{X}_{\mathcal{C}})$ and $f(\mathbf{X}_{\mathcal{H}})$ is the same and represents the class labels predicted by the two models. 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 5.2 and Equation 5.2 it should be that

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

Thus, an equivalent model exists in $\mathcal{H}$ provided that the initial examples are linear independent (*i.e.*, $(\mathbf{X}_{\mathcal{H}}^T \mathbf{X}_{\mathcal{H}})^{-1}$ exists).

Even though the hypothesis learnt in the lower order space can be learnt in the higher order space, there is still merit in carrying out feature reduction because of the following argument. It has been shown that the bounds for error, EP_{err}, in Support Vector machines are defined by the formula [WMC$^+$00]

$$EP_{err} \leq \frac{1}{l} E \left\{ \frac{R^2}{M^2} \right\}$$

where R is the radius of the sphere containing all the transformations of the examples, M is the maximal margin of separation and E is the error of the classsifier. Note that the radius is computed on the transformed space $\mathcal{H}$ *i.e.*, in case of the polynomial kernel it will be calculated in the expanded space. Therefore choosing the feature space to operate is a trade off between achieving the maximum separation M in the input examples and not adding too many redundant dimensions so that the value of R goes up. This problem is even more critical in the case of conjunctive attributes as the dimensionality of the transformed space grows exponentially as the order of conjunction increases. Hence, though we do not gain in expressibility of the model by operating in the pruned feature space, we stand to gain by substantially tightening the bounds on the error.

To get a better understanding of this error bound and its relationship with the number of features we ran some experiments on synthetic datasets, where we studied the effect of increasing the dimensionality on the accuracy of the SVM classifier. The details of the experiment and the results are presented in Section 6.3.1.

6 Experimental Evaluation

In this section we experimentally evaluated the different composite feature selection techniques presented in Section 4. We first describe the datasets and the methodology of our experiments. We then present the results for different schemes in conjunction of two classifiers, C4.5 and SVM. Lastly we compare our schemes with respect to other schemes which use composite features with the help variety of different metrics.

6.1 Dataset Characteristics & Methodology

To evaluate the performance of different selection strategies, we conducted experiments on the UCI datasets [MM98]. Table 1 displays the different characteristics of each of the datasets. There are essentially four possible classification schemes based on different combinations of our composite feature selection strategies. First, no duplicate elimination, no discriminatory feature selection, referred as ***ND-NS***. Second, duplication elimination using identical class distribution, but no discriminatory feature selection, referred as ***IC-NS***. Third, duplicate elimination and using confidence for discriminatory feature selection, referred as ***IC-Co***. Lastly, duplicate elimination and using *j-measure* for discriminatory feature selection, referred as ***IC-JM***. Besides these four selection strategies we also ran the two classifiers, C4.5 and SVM directly on the UCI datasets.

Dataset	No. Attributes		*Number*	*N*umber
	Cont.	Disc.	*C*lasses	*E*xamples
anneal	6	32	6	898
austra	6	8	2	690
breast	10	0	2	699
cleve	6	7	2	303
crx	6	9	2	690
diabetes	8	0	2	768
german	7	13	2	1000
glass	9	0	7	214
heart	13	0	2	270
hepati	6	13	2	155
horse	7	15	2	368
iris	4	0	3	150
labor	8	8	2	57
led7	0	7	10	3200
lymph	0	18	4	148
pima	8	0	2	768
tic-tac	0	9	2	958
wine	13	0	3	178
zoo	0	16	7	101

Table 1: UCI dataset statistics.

We performed our experiments using a 10 way cross validation scheme and computed average accuracy across different runs. We ran our experiments using 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` we used the support threshold of 5.0%. This was done to ensure that the composite features generated are statistically significant. For decision tree classification we use a modified version of c4.5 classifier [Qui93] that can handle sparse datasets. Similarly for SVM classification we use SVMLight [Joa99] classifier with radial basis function kernel.

6.2 Using C4.5 Classifier

Table 2 displays the accuracy values and the number of features selected for different schemes when used in conjunction with a c4.5 classifier. The first column, *Direct*, displays the results obtained by directly running c4.5 on these datasets. The remaining columns displays the accuracy and the number of composite features selected for different classification schemes. The average accuracy is displayed at the bottom of the table. This table does not display the scheme in which no duplicate pruning and no selection *ND-NS*, because the sparse c4.5 classifier is unable to handle the large number of composite features; which are generated as a result of no feature selection.

Dataset	*Direct*		*IC-NS*		*IC-Co*		*IC-JM*	
	Acc.	*# Attr.*	*Acc.*	*# CF*	*Acc.*	*# CF*	*Acc.*	*# CF*
anneal	94.76	38	97.44	8391	97.78	776	96.77	346
austra	85.22	14	84.64	133,218	82.75	6,096	83.77	1,929
breast	95.42	10	95.57	6,779	95.57	412	95.71	76
cleve	80.17	13	79.88	21,718	80.19	1,505	78.19	379
crx	84.93	15	84.06	184,472	84.35	8,447	83.91	2,641
diabetes	76.18	8	75.39	823	76.96	109	77.35	54
german	72.70	20	66.40	196,190	67.70	16,141	69.50	12,749
glass	65.97	9	73.96	731	75.35	178	76.71	63
heart	80.00	13	80.00	6,189	79.63	493	81.85	162
hepati	83.25	19	85.71	53,797	82.50	2,822	83.08	1,808
horse	82.92	22	81.27	47,315	80.97	4,204	80.38	2,492
iris	95.33	4	93.33	80	94.00	26	94.00	14
labor	79.00	16	87.67	1,134	89.33	193	87.33	91
led7	72.88	7	73.84	1,707	73.09	94	73.91	17
lymph	79.72	18	79.00	35,086	74.24	3,469	72.19	2,517
pima	74.22	8	77.61	781	77.09	100	77.34	49
tic-tac	98.64	9	98.02	17,021	97.49	3,343	97.91	2,546
wine	92.75	13	94.93	20,465	94.41	999	94.93	342
zoo	92.09	16	93.27	7,914	95.18	936	94.09	382
Avg.	82.95		83.52		83.41		83.44	

Table 2: Results by using C4.5 Classifier.

After studying the Table 2 we can make the following observations. First, both the *confidence* and *j-measure* selection schemes eliminate a large number of composite features as compared the *ND-NS* scheme. Some times the reduction is up to two orders of magnitude. Of the two selection schemes, *j-measure* is far more selective in picking composite features. However if we compare the average accuracy we find that the all the selection schemes outperform the traditional c4.5 approach, albeit by a small margin. Amongst the different selection schemes, the schemes with only duplicate elimination *ND-NS* performs the best.

6.3 Using SVM Classifier

Table 3 displays the results obtained by the SVM classifier. The first column displays the accuracy values obtained by directly using the SVM classifier and the remaining columns display the accuracy values and the number of composite features obtained by different feature selection schemes outlined in Section 6.1. The last row displays the average accuracy values for different classification schemes.

After studying the results Table 3 we can make the following observations. First, as mentioned in Section 6.2 the selection schemes result in a drastic reduction of composite features given to the classifier. However in the case of SVM classifier the performance of direct classifier is quite comparable to the other feature selection schemes, in fact only two schemes *IC-NS* and *IC-CO* have average accuracy more than that of the direct SVM Classifier.

6.3.1 Evaluation using Synthetic Dataset

To get the better understanding of the Equation 5.2 and the effect of dimensionality on the error bound we conducted some experiments by generating synthetic datasets. To keep the analysis simple the synthetic dataset contains just

Dataset	*Direct*		*ND-NS*		*IC-NS*		*IC-CO*		*IC-JM*	
	Acc.	*# CF*	*Acc.*	*# CF*	*Acc.*	*# CF*	*Acc.*	*# CF*	*Acc.*	*# CF*
anneal	98.33	126	98.44	850,419	97.88	8,391	98.33	776	99.00	346
austra	85.80	51	86.38	616,885	86.96	133,218	86.09	6,096	85.80	1,929
breast	97.14	30	97.00	12,409	96.86	6,779	96.71	,412	97.29	76
cleve	84.11	28	84.44	65,433	83.47	21,718	83.46	1,505	83.45	379
crx	86.09	55	86.67	1,154,453	86.52	184,472	86.09	8,447	84.49	2,641
diabetes	77.35	16	78.14	1,055	78.53	823	78.53	,109	77.87	54
german	75.70	61	71.90	667,263	72.30	196,190	72.60	6,141	74.30	12,749
glass	75.33	21	78.61	2,638	78.12	731	75.28	,178	73.46	63
heart	82.96	19	85.56	10,827	85.56	6,189	84.81	,493	82.96	162
hepati	84.38	34	79.29	1,170,213	81.21	53,797	81.21	2,822	85.79	1,808
horse	85.61	62	82.61	390,272	84.79	47,315	83.43	4,204	82.32	2,492
iris	93.33	13	94.00	,129	94.00	80	94.00	26	93.33	14
labor	89.33	30	77.33	14,182	94.67	1,134	94.67	193	94.67	91
led7	72.41	15	71.25	1,911	72.19	1,707	73.03	94	72.78	17
lymph	84.38	51	81.05	2,040,379	80.33	35,086	81.67	3,469	80.38	2,517
pima	77.34	16	78.77	1,041	79.04	781	78.52	100	78.52	49
tic-tac	95.41	28	98.54	21,368	98.54	17,021	96.97	3,343	97.70	2,546
wine	99.44	38	98.86	526,707	98.30	20,465	99.44	999	98.86	342
zoo	96.00	37	97.00	1,398,654	92.09	7,914	96.00	936	96.00	382
Average	85.61		85.57		85.68		85.67		85.58	

Table 3: Results by using SVM Classifier

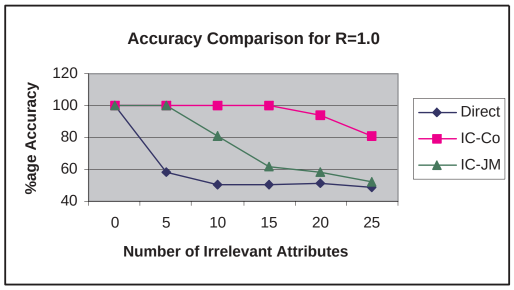
two classes. The attributes making up the dataset are of two kinds, relevant and irrelevant. The relevant attributes influence the class label, whereas irrelevant don't, and can be thought of as noise. For our experiments we restricted the number of relevant attributes to three and varied the number 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.

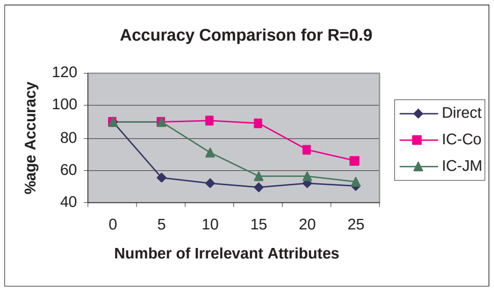
The number of non-relevant attributes control the dimensionality of the problem. To make the dataset realistic we also added some noise to the classification function, the amount of noise is controlled by a parameter called *R value*, R refers to the randomness. While assigning the class label to an 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. Though *R value* does not change the dimensionality of the problem it makes the dataset more realistic.

In our experiments we build a direct SVM classifier using polynomial kernel with the *d* (maximum dimensionality) = 3. Similarly, we run of classification scheme in which the maximum size of the composite feature is restricted to 3. The support threshold for FIDA is set to 1.00 %, this ensures that all possible composite features of length 3 are discovered. In our experiments we vary the *R value* from 1.0, meaning no noise, to *R value* of 0.7, we also vary the number of irrelevant attributes, starting from 0 indicating no irrelevant attributes to 25 irrelevant attributes. The number of irrelevant attributes control the dimensionality of the problem, To keep the discussion simple we only use *IC-CO* and *IC-JM* selection schemes.

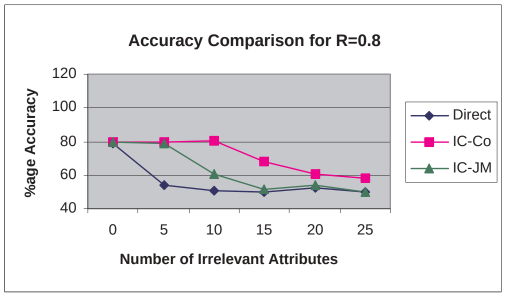
The results of these experiments are shown in Figure 4. The figure contains four graphs for different *R values*,



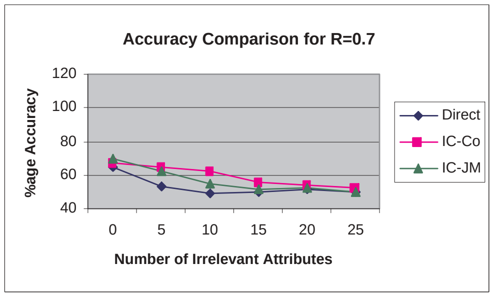
(a) Accuracy Values for R=1.0



(b) Accuracy Values for R=0.9



(c) Accuracy Values for R=0.8



(d) Accuracy Values for R=0.7

Figure 4: Accuracy values for Direct, *IC-Co*, *IC-JM* classifications schemes across varying number of irrelevant attributes and different values of R.

each graph contains three lines corresponding to the three classification schemes, *Direct*, *IC-CO* and *IC-JM*. The X-axis represents the number of irrelevant attributes and the Y-axis represents the accuracy obtained. After studying the Figure 4, we can make couple of inferences. First, the accuracy of the *Direct* scheme is greatly influenced by the number of irrelevant attributes, if we study the Figure 4 (a) we observe that the the *Direct* scheme which starts off with the accuracy of 100% (*i.e*, with no irrelevant attributes) drops sharply to 50% as the number of irrelevant attributes is increased to 10. On the contrary the composite features based schemes, especially the scheme *IC-CO*, show considerable resilience against the number of irrelevant attributes. The accuracy of *IC-CO* drops only to 80% when 25 irrelevant attributes are added. Similarly composite feature based schemes are also more resilient to the noise. Second, we observe that the *IC-CO* schemes outperforms the *IM-JM* scheme, this could be because the *j-measure* scheme is more selective about selecting composite features and hence could result in pruning of using features

6.4 Overall Comparison of Schemes

In this section we compare the accuracy values of all our schemes with the native classifier as well as two other schemes based on composite features, namely CBA [LHM98] and CMAR [LHP01]. To compare the classification schemes across different datasets we use variety of criteria. Each of these criterion assign a single numeric value for each classification scheme, *i.e.*, the criteria function assigns a sing value to every accuracy column in Table 4. Once the criteria has been computed we can compare different classification schemes by comparing the value of this criteria function. We have use four criteria 1) *Average Accuracy:* This is computed by taking mean of accuracy across different datasets for each classification scheme, though this metric is easy to understand, it is biased by the magnitude of the accuracy. 2) *Average Deficiency:* We first compute the maximum attainable accuracy for each dataset across the different classification schemes. Then for each dataset we compute the deficiency, which is one minus the ratio of accuracy attained and the maximum attainable accuracy for that dataset, the average deficiency is the average of over all datasets, ideally we would like this value to be as close to 0.0 as possible, implying the accuracy is equal the maximum attainable accuracy. 3) *Average Rank:* This is a non parametric metric, we first compute the rank on each dataset across different classification schemes (position in the ordered sequence) for each dataset, the average rank is the average of all ranks, this value should be as low as possible. 4) *Number of Max Datasets:* This metric computes the datasets for which a particular scheme achieves the maximum accuracy. Again we would like this value to be as high as possible.

Dataset	*CBA*	*CMAR*	*C4.5*				*SVM*				
			Direct	*IC-NS*	*IC-CO*	*IC-JM*	*Direct*	*ND-NS*	*IC-NS*	*IC-CO*	*IC-JM*
anneal	97.90	97.30	94.76	97.44	97.78	96.77	98.33	98.44	97.89	98.33	99.00
austra	84.90	86.10	85.22	84.64	82.76	83.77	85.80	86.38	86.96	86.09	85.80
breast	96.30	96.40	95.42	95.56	95.57	95.71	97.14	97.00	96.86	96.71	97.29
cleve	82.80	82.20	80.17	79.88	80.19	78.19	84.11	84.44	83.47	83.46	83.45
crx	84.70	84.90	84.93	84.06	84.35	83.92	86.09	86.67	86.52	86.09	84.50
diabetes	74.50	75.80	76.18	75.39	76.96	77.35	77.35	78.14	78.53	78.53	77.88
german	73.40	74.90	72.70	66.40	67.70	69.50	75.70	71.90	72.30	72.60	74.30
glass	73.90	70.10	65.97	73.96	75.35	76.71	75.33	78.61	78.12	75.28	73.46
heart	81.90	82.20	80.00	80.00	79.63	81.85	82.96	85.56	85.56	84.82	82.96
hepati	81.80	80.50	83.25	85.71	82.50	83.08	84.38	79.29	81.21	81.21	85.79
horse	82.10	82.60	82.92	81.27	80.97	80.38	85.61	82.61	84.79	83.43	82.32
iris	94.70	94.00	95.33	93.33	94.00	94.00	93.33	94.00	94.00	94.00	93.33
labor	86.30	89.70	79.00	87.67	89.33	87.33	89.33	77.33	94.67	94.67	94.67
led7	71.90	72.50	72.88	73.84	73.09	73.91	72.41	71.25	72.19	73.03	72.78
lymph	77.80	83.10	79.72	79.00	74.24	72.19	84.38	81.05	80.33	81.67	80.38
pima	72.90	75.10	74.22	77.61	77.09	77.35	77.34	78.78	79.04	78.52	78.52
tic-tac	99.60	99.20	98.64	98.02	97.50	97.91	95.41	98.54	98.54	96.97	97.70
wine	95.00	95.00	92.75	94.93	94.41	94.93	99.44	98.86	98.30	99.44	98.86
zoo	96.80	97.10	92.09	93.27	95.18	94.09	96.00	97.00	92.09	96.00	96.00
Average	83.90	84.38	82.95	83.52	83.41	83.44	85.61	85.57	**85.68**	85.67	85.58
Avg. Def.	3.71	3.14	4.67	4.12	4.24	4.16	1.62	2.50	**1.56**	1.59	1.70
Avg. Rank	6.90	5.80	7.15	7.95	7.75	7.55	4.30	4.50	4.05	**4.00**	4.55
# Max Acc	1	1	1	0	0	1	**4**	3	3	3	**4**

Table 4: Overall comparison of accuracy.

Table 4 displays the accuracy values, as mentioned earlier the ideal scheme should have a large value for the metrics and *Average* and *# Maximum Accurate* and as low value for *Average Rank* and *Average Deficiency*. The maximum values for each of the metrics are displayed in bold, it can be seen that each metric chooses a different scheme as its best. However we can clearly infer that the SVM based schemes outperform other schemes, amongst svm based schemes the results are extremely close with the *I*S-JM having a slight edge.

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.

References

[AAP00] Ramesh Agrawal, Charu 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.

[AIS93] 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.

[BU92] Carla E. Brodley and Paul E. Utgoff. Multivariate versus univariate dcision trees. Technical report, University of Massachusetts, 1992.

[DKS95] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretisation of continuous features. In *Machine Learning: Proceedings of the Twelfth Internation Conference*, 1995.

[FI93] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, 1993.

[GD90] Pagallo G. and Hassler D. Boolean feature discovery in empirical learning. *Machine Learning*, 1990.

[HPY00] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *Proc. 2000 ACM-SIGMOD Int. Conf. on Management of Data (SIGMOD'00), Dallas, TX*, May 2000.

[Joa99] T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press, 1999.

[LHM98] Bing Liu, Wynne Hsu, and Yiming Ma. Integrating classification and association rule mining. In *4th Internation Conference on Knowledge Discovery and Data Mining*, 1998.

[LHP01] Wenmin Li, Jiawei Han, and Jian 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.

[LZO99] Neal Lesh, Mohammed J. Zaki, and Mitsunari Ogihara. Mining features for sequence classification. In *5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1999.

[MM98] C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1998.

[MP91] Patrick M. Murphy and Michael J. Pazzani. Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. In *Proc. of the 8th Int Ẇorkshop on Machine Learning*, 1991.

[MR89] C. J. Matheus and L.A. Rendell. Constructive induction on decision trees. In *Proceedings of the Eleventh International Joint Conference on Artifical Intelligence*, 1989.

[Qui93] J. Ross Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[S.78] Michalski R. S. Pattern recognition as knowledge guided computer induction. Technical report, University of Illinois at Urbana Champaign, 1978.

[SG92] 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.

[SK01] Masakazu Seno and George 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.

[Vap98] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[WMC$^+$00] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. Feature selection fof svms. *Advances in Neural Information Processing Systems*, 2000.

[Zak00] Mohammed Javeed Zaki. Scalable algorithms for association mining. *Knowledge and Data Engineering*, 12(2):372–390, 2000.

[Zhe00] Zijian Zheng. Constructing conjunctive attributes using production rules. *Journal of Research and Practice in Information Technology*, 2000.

[Zij96] Zheng 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.

[ZLM00] Mohamed J. Zaki, Neal Lesh, and Ogihara Mitsunari. Planmine: Predicting plan failures using sequence mining. *Intelligence Review, special issue on the Application of Data Mining*, 2000.