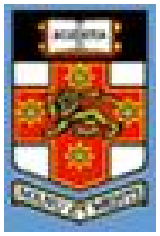
IC^2: An Interval based Characteristic Concept Learner

Pramod K. Singh
Artificial Intelligence Group
School of Computer Science and Engineering
University of New South Wales, Sydney, NSW - 2052, Australia
Email: pksingh@cse.unsw.edu.au

UNSW-CSE-TR-0442
January 2005

University of New South Wales



School of Computer Science and Engineering
University of New South Wales
Sydney, NSW - 2052, Australia

ABSTRACT

In many real-world problems it can be argued that classification based on characteristic descriptions delivers a more correct and comprehensive concept than the existing discriminative descriptions. Most classification algorithms suffer from an inability to detect instances of classes which are not present in the training set. A novel approach for characteristic concept rule learning called IC^2 is proposed in this paper. It uses the descriptions of one class objects for learning concept rules for the classification and descrimination of unseen instances. In this approach interval-based class characteristic rules are induced from the descriptions of the labeled objects of a class. The paper illustrates the approach and presents the empirical results obtained on some data sets of continuous feature variables. The IC^2 classifier is evaluated and its classification accuracy is compared with a state-of-the-art characteristic concept classification algorithm ID3-SD as proposed by Davidsson [2].

1 Introduction

Concept learning from the examples is a process of inducing descriptions of a set of objects, which are classified by a human expert or a system as instances of a meaningful class. The learned concept is supposed to help in determining the class of other unseen instances. There are two types of distinguished description methods in the literature (1) characteristic descriptions and (2) discriminative descriptions. A characteristic description of a class of objects describes the sufficient conditions for class membership and enables a system to identify all instances of the class and reject all instances of the other (disjoint) classes. On the other hand, a discriminative description discriminates between instances of one class and the instances of a pre-defined set of other classes. A discriminative description needs only to specify the properties relevant in the context of a fixed set of the other classes.

Most learning systems, which learn from examples, deliver discriminant description and ignore the problem of the classification of an instance of an unknown category. Unfortunately, in many application domains it cannot be assumed that the collected examples in the training set represent all relevant categories because the number of these categories can be quite large. Also sometimes it is practically impossible to collect data of all classes at the time of description learning. There is likely to be misclassification of instances of novel or unknown classes by a discriminative classifier unless the training data is fully comprehensive. In some problem domain e.g. medical diagnosis, the misclassification of any instance may cost a lot. What we need under such situations is the ability to reject instances of the categories that the system has not been trained on. For example, consider the decision mechanism of disease diagnosis, where the cost of misclassification of a disease D_1 (e.g. a lung disease) is too high. If we use discriminative description concept, then we need to collect examples of disease D_1 and present them as positive examples and descriptions of all other diseases D_i (i $\neq$ 1), which might occur in the application domain as negative examples (e.g. all lung diseases and all other diseases sharing the symptoms with the lung disease D_1). Otherwise, it could happen that the induced disease description may misclassify a non-disease instance as an instance of the disease. In such a situation, it is better to remain silent rather provide an incorrect diagnosis.

2 Related Work

Several fields have contributed to the task of characteristic concept learning. Primarily characteristic concept learning can be seen as a problem of one class classification. There are several algorithms for one class classification based on SVMs (Support Vector Machines) proposed in the literature [5]-[7]. In these approaches one has to select specific methods of data representation and a kernel for a class. Surprisingly the classifiers induced by such SVMs are also quite sensitive to the tuning of the parameters and this is not transparent. Some algorithms using an ILP (Inductive Logic Programming) approach have also been proposed [9],[10]. The hypothesis induced by these algorithms are dependent on specific heuristics developed for a class. Besides, the ILP based algorithms lead to the complexity

problems.

Considering it as a descriptive generalization problem, Han and Fu [3] introduced an attribute-oriented induction algorithm for data generalization in data mining applications, but in their framework, background knowledge such as taxonomies is needed for generalizing data and objects. Smyth and Mellstrom [4] presented an algorithm for the novel class detection using an instance-based method, but the main limitation of the algorithm is its inability to produce the explicit rules, desired in many applications. Holte et al. [1] have shown that the CN2 algorithm can be modified to learn characteristic descriptions in the form of rule-based maximum specific descriptions. Constructing maximum specific descriptions is the simplest way of learning characteristic concepts; for each category one needs only to calculate the minimum and maximum value of each feature from the training instances belonging to the category. Hence it learns the most specific descriptions possible. On the other end, some learners of the AQ family e.g. AQ11 [8] and AQ15c [14] are able to learn the most general descriptions which in some cases degenerate into discriminative concepts. In most applications these extreme approaches are inadequate due to being static and having no scope of controlling the degree of generalization. To overcome the problem of static generalization an algorithm called ID3-SD, an extension of ID3 algorithm of Quinlan [13], was proposed by Davidsson [2].

Davidsson has shown some competitive classification results using ID3-SD on coin sorting and some other data sets. However, it has some constraints: firstly, it makes an assumption that the distribution of feature values in a class follows a normal distribution. Secondly, it needs examples of other classes as negative data to select the features and their class cut points. The feature selection in ID3-SD algorithm, using information gain or gain ratio, is quite dependent on the negative examples in the training data set. In this paper we propose an interval based algorithm for learning characteristic concepts using positive only examples and synthetically generated uniform data as negative examples for each class. The algorithm uses a wrapper method for feature selection.

The rest of the paper is organized in the following manner. The Interval rule concept is introduced in Section IV. The algorithm for characteristic concept classifier IC^2 is described in Section V. Experimental results is presented in Section VI and the paper is concluded in Section VII.

3 Problem Statement

The characteristic concept learner we are interested in can be defined as follows. Suppose there is a set of objects $O = \{o_1, o_2, ..., o_n\}$ of a class, say C. Each object o_i of the class is described by a set of features $F = \{f^1, f^2, ..., f^m\}$, where $f^1..f^m$ are the features representing the nature of an object. A set of characteristic concept rules $R_C = \{R_1, R_2, .., R_p\}$ is learned for the class C in order to classify the unknown instances. The rules in R_C are mutually exclusive. Note that the rule set is defined on the same feature set F. A rule may use a single or multiple features from the feature set.

Let a new unknown instance I be described by the feature set F (with the values of

features $f^1..f^m$) which is classified as a member of the class C using the rules of R_C on F. In case the feature values of F are not able to satisfy the conditions of any rule of the rule set R_C, the instance I is regarded as a member of a novel class by the system.

4 Interval Rule Concept

Interval concept we are proposing here is defined as a set of rules, where each rule has a conjunct of conditions for a feature set. A condition is a numeric value range also called an interval limit with a lower bound and an upper bound for a numerical feature or a single value for a categorical feature. The interval limits for a numerical feature of a class is created by decomposing the range of values of that feature in a given dataset into k intervals. The value of k is determined separately for each feature of the class using statistical method. Out of k intervals, if any two adjacent intervals are having non zero allocation of instances would be merged to produce a larger size interval.

4.1 Simple Interval Rules

Simple interval rules are defined on single feature of the data set.

Definition 4.1 *Let feature f of the class C with values $f_1...f_n$, where n is the number of objects in class C. The minimum and maximum values of the feature f in the data set are:*

$$f_{min} = Min(f_i) \tag{1}$$

$$f_{max} = Max(f_i) \tag{2}$$

Further, the mean value f_{mean} and the standard deviation f_{SD} of the feature f are calculated.

$$f_{mean} = \frac{\sum_i f_i}{n} \tag{3}$$

$$f_{SD} = \sqrt{\frac{\sum (f_i - f_{mean})^2}{n-1}} \tag{4}$$

where i = 1..n. The number of intervals k is calculated as follows:

$$k = \frac{(f_{max} + \frac{f_{SD}}{2}) - (f_{min} - \frac{f_{SD}}{2})}{\sigma * f_{SD}} \tag{5}$$

In (5) f_{SD} determines the number of k and width of range in each interval. The motive of using f_{SD} in interval determination is that if a feature is having larger variation, should have finely divided range otherwise larger range width is produced. Here σ is specificity factor for increasing the effect of f_{SD} and generated algorithmically. The specificity of the rules increases monotonically with the increment of σ. σ is initialized by one in the algorithm.

Definition 4.2 *Let the values of feature f be divided into k intervals and number of the objects in class C be n. The objects are allocated to the k intervals based on the respective values of the feature f for each object. Suppose O_j indicate the number of objects allocated to the j^{th} interval, where j = 1..k. The operators of interval merger $Merg$ for obtaining the disjunct interval rules are defined as:*

$k_{merg} = Merg(k_j, k_{j+1})$, *if* O_j *and* O_{j+1} *are non zero*

where k_{merg} is an interval produced after merger of the intervals k_j and k_{j+1}. Note that k_j and k_{j+1} are adjacent intervals and O_j and O_{j+1} are the numbers of allocated objects in these intervals, respectively. Intervals with zero allocation are ignored.

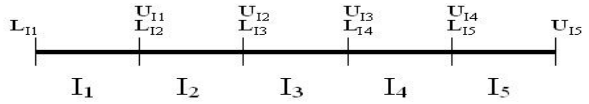
Corollary 4.3 *Let k be the number of intervals generated initially from the training data set and k_{rules} be the number of disjunct interval rules on the feature f. Then the inequality condition $k_{rules} <= \frac{k}{2} + 1$ holds as in the worst case scenario every second interval is empty.*

Example 1 *Suppose I_1, I_2, I_3, I_4, and I_5 are the intervals generated in a feature by (5), Fig. 1(a). These intervals are in order and values of limits are also in increasing order from first interval to the last. It implies that the lower limit of one interval will always be less than the lower limit of its successor interval. Let L_{I1}, L_{I2}, L_{I3}, L_{I4} and L_{I5} be lower limits of intervals I_1, I_2, I_3, I_4, and I_5, respectively, and U_{I1}, U_{I2}, U_{I3}, U_{I4}, U_{I5} be their upper limits. Except for the first interval the lower limit of an interval is the upper limit of its predecessor. Thus the lower limit L_{I2} of I_2 and upper limit U_{I1} of I_1 has same value. Further, assume training examples allocated to the intervals I_1, I_2, I_3, I_4, and I_5 are 4, 2, 0, 0, and 1 respectively. The final disjunct interval rules produced from $Definition$ 4.2 would be two only, first after joining I_1 and I_2 as single interval be I_{12} and I_5 as an independent second interval, shown in Fig. 1(b). The lower limit of interval I_{12} is L_{I1} and upper limit is U_{I2} due to the merger of intervals I_1 and I_2. Clearly intervals I_3 and I_4 are ignored in rule production.*

4.2 Composite Interval Rules

The composite interval rules are the rules defined using more than one feature by applying cartesian products among the simple interval rules derived for each feature of the feature set used in the composite interval rules definition.

Definition 4.4 *Let features f1 and f2 be involved in the definition of the classifier and assume that the set of disjunct interval rules produced using the simple interval rules method defined above for f1 and f2 to be S(f1) and S(f2), respectively. Then the composite interval rules are the rules produced by the cartesian product operator $Cart$ on S(f1) and S(f2).*



(a)

(b)

Figure 1: Single Feature Interval Representation (a) Initial intervals created (b) Final disjunct interval rules (see example 1)

Furthermore, the created cartesian rule set C_{rules} is used for the allocation of training objects. Finally the reduced cartesian rule set C^r_{rules} is defined as follows:

$$C_{rules} = Cart(S(f1), S(f2)) \tag{6}$$

$$C^r_{rules} = \{\forall rule | rule \in C_{rules} \wedge cover(rule) \neq 0\} \tag{7}$$

where $cover(rule)$ is the number of objects covered by the rule.

Example 2 *Suppose the features a and b of a data set are used for the classification. Assume that the feature a has two disjunct interval rules I_{a1} and I_{a2}, and feature b also has two disjunct interval rules, say I_{b1} and I_{b2}, as shown in Fig. 2. The cartesian product of the rules of feature a and b produces four sets of disjunct composite interval rules. Let us say the rules produced by cartesian operator are $I_{a1}I_{b1}$, $I_{a1}I_{b2}$, $I_{a2}I_{b1}$ and $I_{a2}I_{b2}$. As shown in Fig. 2 the rule $I_{a2}I_{b2}$ does not cover any instance of the data set. Thus the reduced composite interval rules for the data set are three only. For simplicity, we presented here an example of two feature based composite interval rules. However, the method can be applied to any number of features. But the complexity of rule conditions and the number of rules increases exponentially with the number of features in composite interval rules creation.*

5 IC^2 Generation Algorithm

The IC^2 generation algorithm consists two main steps - rule generation and feature selection. In the first step a function called **Interval_Rules** creates concept rules based on the definitions explained in the section III. It works on single features for simple rules and generates

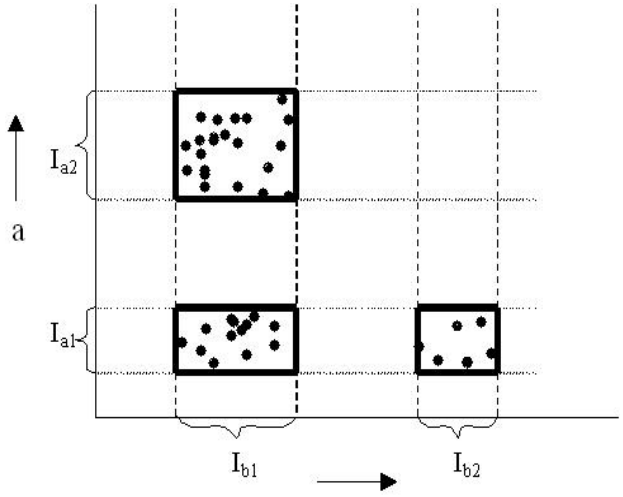


Figure 2: Cartesian Product Rules of Multiple Feature Intervals (see example 2)

composite rules using multiple features. The IC^2 classifier generated using all of the features of the data set, produces a very specified characteristic concept and defies the objective of concept generalization. Therefore, for generalization of concept learning the feature selection step is quite important. A feature selection method based on a *wrapper model* [11] is used in this algorithm. The function **Feature_Select** uses a goodness measure for selection of promising features.

For measuring the goodness of the interval rules generated from a feature or a set of features we use two measures called *precision* and *recall* of the rules. *Precision* is the ratio of the number of target class examples correctly covered by the rules to the total number of examples (including target class and negative examples) covered. *Recall* of the rules is ratio of the number of target class examples covered by rules to the total number of target class examples. An accuracy factor $f\_accuracy$ measure is also used in our feature selection algorithm, which combines *recall* and *precision* statistics. A feature or feature set producing interval rules with a larger $f\_accuracy$ is preferred. $f\_accuracy$ is used to rank the features and to guide the specificity factor σ in the characteristic interval rules generation process. $f\_accuracy$ for the rules generated on a feature or feature set is calculated using following formula:

$$f\_accuracy = \frac{\beta + 1}{\frac{\beta}{recall} + \frac{1}{precision}} \tag{8}$$

where β is a generalization parameter. $f\_accuracy$, with a value of β greater than 1, favors *recall* and more generalization of concepts, whereas with a value smaller than 1, it favors *precision* and more specific rules.

The goodness measure of interval rules needs negative examples for computation of the *precision* of rules. As pointed out by Mitchell [12] that the intended machine learner needs

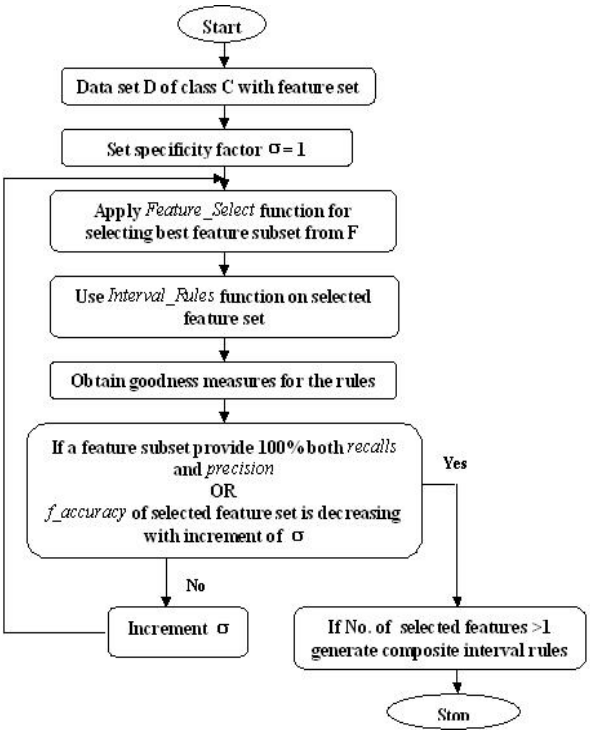


Figure 3: IC^2 Generation Algorithm

some prior information on possible non-uniform probability distribution functions, that we are looking for in the target class data, can be derived from a uniform distribution of the class population. Moreover, a key intuition in machine learning is that similarly valued data points have similar classes. Hence we can say that the data is more uniform if it is less likely to come from any other smooth, simple probability distribution function. Based on Mitchell's concept we generated uniformally distributed negative examples for each target class to be learned. Unlike Mitchell's $PolarC4.5$ and $Polar$ algorithms, where a uniform distribution for negative data is used for getting split points, Interval learner uses the generated negative examples for testing the rules coverage only. Thus the dependency of learner is not as much on negative data as in decision tree based algorithms.

For selecting the set of features we use the popular forward sequential selection (FSS) method in our algorithm. For rule definition the algorithm starts with the most general rules, which cover the entire instance space. It conducts a depth first general-to-specific search. A rule is specialized by increasing the interval specificity factor σ which is initialized by one. The specialization using the specificity factor increases the lower bound and decreases the upper bound of the rule conditions. The effect of specificity factor is evaluated using $f\_accuracy$. Fig.3 summarizes the IC^2 generation algorithm; for ease of exposition we have omitted various details from the presentation. However, the details of two main functions used in the algorithm are presented below in Table I and II.

TABLE I: INTERVAL_RULES FUNCTION FOR INTERVAL RULES GENERATION

Simple Interval_Rules

Input: Training Dataset D, feature f.
Output: Disjunct Interval Set.
S_Interval Algorithm:

Get min and max values of f in D.
Compute $mean$, SD and no. of intervals k.
Compute lower and upper bounds for each interval k.
Allocate instance $i \in D$ in one of the intervals k
    based on value of f.
Merge adjacent intervals of non-zero coverage
    and define new bounds.
Delete intervals of zero coverage.
Return Disjunct intervals.

Composite Interval_Rules

Input: Training Dataset D, feature set F.
Output: Cartesian Interval Set.
Initialization: Cart-intervals = ϕ.
C_Interval Algorithm:

For each feature f ∈ F do
{
//operator X is cartesian product
        Cart-interval = Cart-intervals X S_interval(f).
}
Allocate instance $i \in D$ to one of the rule ∈ Cart-intrvals.
Remove rules ∈ Cart-intervals have zero instance coverage.
Return Cart-intervals.

TABLE II: FEATURE_SELECT FOR FEATURE SELECTION

Feature_Select Function

Input: Dataset D, feature set F.
Output: Feature Subset S.
Initialization: $S = \phi$.
Feature_Select Algorithm:

```
// use forward sequential selection method
While(Recall and Precision both not 100% or
f_accuracy increasing)
{
    For each f ∈ F do
    {
        S = S ∪ f.
        Calculate Goodness(S).
        S = S − f.
    }
    Select best feature f.
    S = S ∪ f.
    F = F - f.
}
Return S.
```

Goodness of a feature

Input: Dataset D, feature set F.
Output: goodness measures.
Goodness Algorithm:

```
Generate uniformally distributed negative dataset U for D.
Divide D evenly into n groups, S_1, S_2, ..., S_n.
For i from 1 to n S_n and U as testing data and rest of
    data as training data
{
    if F has only one element{
    // generate simple interval rules
        Take feature f from F.
        Generate rules by S_Interval(f) using training data.
    }
    else{
    // generate composite interval rules
        Generate rules by C_Interval(F) using training data.
    }
    Test the rules on the testing data.
    Count the total number of test examples correct and
        incorrect.
}
Calculate goodness measures: recall, precision and f_accuracy.
Return goodness measures.
```

Initially, simple interval rules using specificity factor $\sigma = 1$ for each feature of the data set are generated and tested using 10 fold cross validation. For testing we use synthetically generated negative examples for each target class. The data generated in the above operation is used for calculation of *precision*, *recall* and *f_accuracy* measures.

First we select all the features of the maximum *recall* (often 100%). Further we improve the *precision* of rules by creating the composite rules with more than one features or by increasing the specificity factor σ. If a feature produces 100% *recall* and *precision* both, or the *f_accuracy* of the subsequent iteration is decreasing, the process is terminated and the feature set is selected for the characteristic rules generation.

Composite interval rules are generated using cartesian product which increases the complexity in the conditions of the rules as the number of features increases. Therefore, in order to reduce the complexity of rules we introduce a *hybrid interval rule* concept which uses simple interval rules along with composite rules. One can select the number of features from the selected feature set to be used for creating composite rules and rest of the features are used for simple rules. These hybrid rules are evaluated using the values of all features involved in both composite and simple disjunct rules.

6 Experimental Results

We would like to test the characteristic concept classifier developed using the interval method on some real data and compare the performance with other methods. Since there were no characteristic concept rule learner using only positive instances at our disposal, we use the extended version of ID3-SD [2] for our comparison. Similar to $PolarC4.5$ [12], to use ID3-SD algorithm we add uniformally distributed negative data to produce the characteristic concept rules on positive only data. We term the extended algorithm as $PolarID3 - SD$. The generalization factor α used in ID3-SD algorithm in our experiment is for 100% at the value of $\alpha = 0.0001$ in the following eq.(9) of ID3-SD.

$$P(m - \lambda_{\frac{\alpha}{2}}\sigma < x < m + \lambda_{\frac{\alpha}{2}}\sigma) = 1 - \alpha \tag{9}$$

where, x is a normally distributed stochastic variable, m is mean, σ is the standard deviation (SD), and λ is a critical value depending on degree of generalization factor α.

The idea for IC^2 emerged while working on a real world application concerning the problem of disease classification from High Resolution Computed Tomography (HRCT) lung images. In order to evaluate the interval based learner approach it has to be applied to other data sets also. In this section we report the results of experiments conducted on three different domains containing numerical features: the classic IRIS classification problem, a Wine recognition problem, and a HRCT lung image regions classification problem.

In this experiment we simulated one class characteristic learning by taking the data of one category during training and then testing on the instances of all categories available in the dataset. We used 10 trial 10 fold cross validation method. We divided the training class data into 10 random sets, S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9 and S_10. In each iteration, we

pick up one set for testing along with the instances of other categories and remaining nine sets containing only the positive sample and synthetic uniformally distributed negative data are used for training. At best, the algorithm should classify the instances of the category correctly for which the concept was generated and reject all those instances belonging to the other categories.

6.1 IRIS Plant Data

IRIS dataset is one of the most popular datasets from the UCI repository for testing the machine learning algorithms. This dataset contains 3 types of IRIS plants (Setosa, Versicolor and Virginica) of 50 instances of each. The data set has 4 numerical features, sepal length, sepal width, petal length, and petal width. Table III shows the classification results when the algorithm was trained on the instances of one category and tested on the instances of the training category as well as other two categories.

TABLE III: % ACCURACY OF CLASSIFICATION OF IRIS CLASSES

	Iris Setosa	Iris Versicolor	Iris Virginica
PolarID3-SD	93.5±2.7	90.4±1.9	79.3±2.9
IC^2	98.7±1.2	89.3±2.8	86.6±3.7

6.2 Wine Data

Wine data set is provided by Institute of Pharmaceutical and Food Analysis and Technicologies in Genoa and available at the UCI repository. This dataset contains instances of three categories. The categories of wine depends on the fermentation yeast and technique used for preparation. The datset contains the chemical analysis of these wines in the form of 13 numerical attributes. The dataset consists of 59 instances of wine of type 1, 71 of type 2, and 48 of type 3. Table IV shows the results of classification on this dataset.

TABLE IV: % ACCURACY OF CLASSIFICATION OF WINE CLASSES

	Wine 1	Wine 2	Wine3
PolarID3-SD	89.5±3.1	92.9±2.7	88.2±1.9
IC^2	95.2±1.1	97.3±0.7	97.5±2.1

6.3 HRCT Image Data

This dataset is generated from the HRCT lung images by applying the statistical operator to Discrete Cosine Transform (DCT) of the labelled regions. The regions of the images are labelled by the radiologist as the region of interest (ROI) for the diagnosis of lung diseases. The feature generation technique is reported elsewhere. In our experiment we selected ROIs of 4 lung disease types; Emphysema, Honey Combing (HC), Ground Glass Opacities (GGO), and Consolidation. We collected 63 ROIs of Emphysema, 48 of Honey Combing, 59 of GGO, and 27 of Consolidation. Each ROIs are converted into a vector of 20 numerical features. Thus we have total 197 instances of these four categories. Table V shows the results of the classification of disease ROIs.

TABLE V: % ACCURACY OF CLASSIFICATION OF HRCT LUNG ROIs CLASSES

	Emphysema	HC	GGO	Consolidation
PolarID3-SD	93.2±1.3	87.4±2.4	83.7±1.8	95.9±1.2
IC^2	99.8±0.8	94.71±1.9	90.5±2.2	100±0

The $PolarID3-SD$ is a version of ID3-SD algorithm implemented in our lab. Both $PolarID3-SD$ and IC^2 used uniformally distributed data as proposed by Mitchell [12] for $Polar$. However, IC^2 only uses the data for testing not for generation of rules. In general the IC^2 performed much better than the $PolarID3-SD$ learner.

7 Conclusion

We proposed a characteristic concept classifier based on an interval rule concept using instances of only one class. The algorithm uses a "'wrapper model"' for deciding which feature should be selected for creating simple or composite interval rules. We used a greedy forward sequential search method for selecting the features; however, like hill climbing algorithms it can have a problem with local extrema. Unlike the decision tree based algorithm the proposed algorithm uses uniformaly distributed negative data for the evaluation of the interval rules coverage and selection of most appropriate feature subset for the generalization of the target class concept. This algorithm also follows the popular general-to-specific search framework of interval rule generation and uses *recall* and *precision* measures to determine if a rule set is interesting.

Preliminary empirical comparison with $PolarID3-SD$, an extended version of ID3-SD, indicates that IC^2 classifier compares quite favorably in classification accuracy. It should also noted that our emphasis is not only on accuracy and correctness but also on the generalization methods. The concept descriptions produced by IC^2 are more specific than descriptions produced by systems using discriminant generalization algorithms, because such systems deliver concept descriptions using only descriptors necessary to discriminate between

the instances of the target concepts. Therefore, the descriptions produced by a system based on IC^2 approach may be more comprehensible.

The current version of IC^2 handles numerical features only in the rule induction. In future we will continue to improve this algorithm in order to allow categorical and missing data features. More exeriments will also be conducted on real and simulated data of various application areas.

Acknowledgment

This research work was partially supported by the Australian Research Council through a Linkage grant (No LP0212081), with Medical Imaging Australasia as clinical and Philips Medical Systems as Industrial partners.

References

[1] R.C. Holte, L.E. Acker, and B.W. Porter, *Concept of Learning and the Problem of Small Disjoints*, Proc. Int. Joint Conf. Artificial Intelligence (IJCAI), 1989, pp.813-818.

[2] P. Davidsson, *ID3-SD: An Algorithm for Learning Characteristic Decision Trees by Controlling the Degree of Generalization*, Technical Report No.LU-CS-TR, Department of Computer Science, Lund University, Sweden, 1995.

[3] J. Han, and Y. Fu, *Attribute-oriented induction in data mining*,Advances in Knowledge Discovery and Data Mining, U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (eds.), 1996, pp.399-421.

[4] P. Smyth, and J. Mellstrom, *Detecting Novel Classes with Applications to Fault Diagnosis*, Proc. Ninth Int. Workshop on Machine Learning, 1992, pp.416-425.

[5] M. Manevitz, and M. Yousef, *One-Class SVMs for Document Classification*, Journal of Machine Learning Research, 2001, pp.139-154.

[6] B. Scholkopf, J.C. Platt, J. Shawe-Taylor, A.J Smola, and R.C. Williamson, *Estimating the Support of a High-Dimensional Distribution*, Technical Report, MicroSoft Research, 1999, MSR-TR-99-87.

[7] N.A. Murshed, F. Borrtolozzi, and R. Sabourin, *Classification of Cancerous Cells based on the One-Class Problem Approach*, SPIE Conference on Applications and Science of Artificial Neural Networks II, Orlando (USA), 1996, vol.2760, pp.487-494.

[8] R.S. Michalski, and, J.B. Larson, *Selection of Most Representative Training Examples and Incremental Generation of VL Hypotheses: The Underlying Methodology and the Description of Programs ESEL and AQ11*, Tecnical Report 877, Computer Science Department, University of Illinois, Urbana, 1978.

[9] W. Emde, *Inductive Learning of Characteristic Concept Descriptions*, Proc. Int. Worshop on Inductive Logic Programming, 1994, vol.237, pp.51-70.

[10] E. McCreath, and A. Sharama, *ILP with Noise and Fixed Example Size: A Bayesian Approach*, Proc. Int. Joint Conference on Artificial Intelligence, 1997, vol.2, pp.1310-1315.

[11] G. John, R. Kohavi and K. Pfleger, *Irrelevent Features and the Subset Selection Problem*, Proc. Int. Conf. on Machine Learning, 1994.

[12] A.R. Mitchell, *"Boosting" a Positive-Data-Only Learner*, Int. Conf. on Machine Learning, 2000, pp.607-714.

[13] J.R. Quinlan, *Induction of Decision Trees*, Machine Learning, 1986, vol.1, no.1, pp.607-714.

[14] J. Wnek, K. Kaufman, E. Bloedorn, and R.S. Michalski, *Selective Induction Learning System AQ15c: The Method and User's Guide*, Tecnical Report MLI 95-4, Centre for Machine Learning and Inference, George Mason University, 1995.