

CLASSIFICATION WITH
OVERLAPPING FEATURE
INTERVALS

A THESIS
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

by
Nakime Desai Kog
January, 1995

Q
325.7
.K63
1995

CLASSIFICATION WITH OVERLAPPING FEATURE INTERVALS

A THESIS

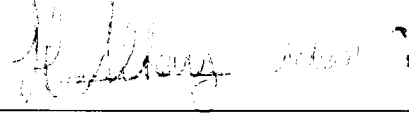
SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING AND INFORMATION SCIENCE
AND THE INSTITUTE OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

by

Hakime Ünsal Koç

January, 1995

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



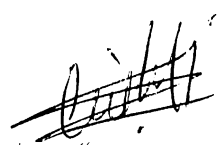
Assoc. Prof. H. Altay Güvenir (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



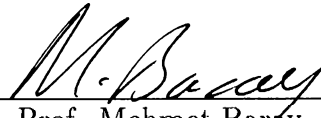
Asst. Prof. Kemal Oflazer

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.



Asst. Prof. İlyas Çiçekli

Approved for the Institute of Engineering and Science:



Prof. Mehmet Baray
Director of the Institute

Q
325.7
-K63
1995

B011007

ABSTRACT

CLASSIFICATION WITH OVERLAPPING FEATURE INTERVALS

Hakime Ünsal Koç

M.S. in Computer Engineering and Information Science

Advisor: Assoc. Prof. H. Altay Güvenir

January, 1995

This thesis presents a new form of exemplar-based learning method, based on overlapping feature intervals. Classification with Overlapping Feature Intervals (COFI) is the particular implementation of this technique. In this incremental, inductive and supervised learning method, the basic unit of the representation is an *interval*. The COFI algorithm learns the projections of the intervals in each class dimension for each feature. An interval is initially a point on a class dimension, then it can be expanded through generalization. No specialization of intervals is done on class dimensions by this algorithm. Classification in the COFI algorithm is based on a majority voting among the local predictions that are made individually by each feature.

Keywords: machine learning, supervised learning, inductive learning, incremental learning, overlapping feature intervals, concept description

ÖZET

ÇAKIŞIK ÖZELLİK ARALIKLARI İLE SINIFLANDIRMA

Hakime Ünsal Koç

Bilgisayar ve Enformatik Mühendisliği, Yüksek Lisans

Danışman: Doç. Dr. H. Altay Güvenir

Ocak 1995

Bu tezde örnek-tabanlı öğrenme için çakışık özellik aralıklarına dayalı yeni bir teknik sunulmuştur. Çakışık Özellik Aralıkları ile Sınıflandırma (COFI) bu yöntemin özel bir uygulamasıdır. Bu çıkarımsal, artımlı ve yönlendirilmiş öğrenme tekniğinde en temel gösterim birimi *aralıktır*. COFI algoritması tüm özelliklere ait her sınıf eksenindeki aralıkların izdüşümünü öğrenir. Aralıklar ilk olarak sınıf eksenlerinde birer noktadrlar, daha sonra tüm bir eksen boyunca genelleştirmeyle genişlerler. Bu algoritmada herhangi bir özelleştirme gerçekleştirilmez. Öğrenme işleminden sonra, tahmin etme işlemi özelliklerin kendi adlarına yaptığı tahminler arasındaki oy çokluğuna dayanır.

Anahtar Sözcükler: öğrenme, yönlendirilmiş öğrenme, çıkarımsal öğrenme, artımlı öğrenme, çakışık özellik aralıkları, sınıf tanımı

ACKNOWLEDGMENTS

I would like to thank all people, especially to my family, who supported and encouraged me throughout this work. I would not have managed to finish this study and reach my goal without unfailing support and help of my family. I would like to thank Anatolian University, Industrial Engineering, and Electrical & Electronics Engineering Departments for providing me comfortable environment for my studies. I would like to express my gratitude to my advisor Assoc. Prof. H. Altay Güvenir for his valuable comments and endless help. Finally, special thanks to my husband, Özgür Koç, who has provided a motivating support and helped me to prepare and follow effective studying principles.

Contents

1	Introduction	1
2	Concept Learning Models	5
2.1	Exemplar-Based Learning	9
2.1.1	Instance-Based Learning (IBL)	10
2.1.2	Nested-Generalized Exemplars (NGE)	13
2.1.3	Classification By Feature Partitioning	16
2.2	Decision Tree Techniques	20
2.2.1	Decision Trees	20
2.2.2	1R System	22
2.3	Statistical Concept Learning	23
2.3.1	Naive Bayesian Classifier (NBC)	25
2.3.2	The Nearest Neighbor Classification	29
3	Classification with Overlapping Feature Intervals	32
3.1	Characteristics of the COFI Algorithm	32
3.1.1	Knowledge Representation Schemes	33

3.1.2	Supervised Learning	34
3.1.3	Incremental Learning	34
3.1.4	Inductive Learning	38
3.1.5	Domain Dependence in Learning	39
3.1.6	Number of Concepts and Number of Features	39
3.1.7	Properties of Feature Values	40
3.2	Description of the COFI Algorithm	40
3.2.1	Training in the COFI Algorithm	41
3.2.2	Prediction in the COFI Algorithm	48
3.2.3	Handling Missing Attribute Values	52
3.2.4	From Intervals to Rules	54
3.3	Comparison of COFI with Other Methods	57
4	Evaluation of the COFI Algorithm	60
4.1	Theoretical Evaluation of the COFI Algorithm	60
4.2	Empirical Evaluation of the COFI Algorithm	62
4.2.1	Methodology of Testing	62
4.2.2	Experiments with Artificial Datasets	63
4.2.3	Experiments with Real-World Datasets	74
5	Conclusions and Future Work	80
A	Concept Descriptions of the Real-World Datasets	89

List of Figures

2.1	Classification of exemplar-based learning algorithms.	10
2.2	An example concept description in the IBL algorithms in a domain with two features.	11
2.3	An example concept description of the EACH Algorithm in a domain with two features.	15
2.4	An example concept description of the CFP Algorithm in a domain with two features.	17
2.5	Constructing segments in CFP by changing the order of the training dataset.	19
2.6	Constructing intervals in the COFI Algorithm with using same dataset as used in Figure 2.5 .	20
2.7	The NBC Algorithm.	26
2.8	Computing the <i>a posteriori</i> probabilities in the NBC Algorithm.	28
2.9	The NN* Algorithm.	30
3.1	An example of construction of the intervals in the COFI Algorithm.	36
3.2	An example of construction of the intervals in the COFI Algorithm with the order of the training instances is changed.	37

3.3	An example of construction of the intervals in the COFI Algorithm.	43
3.4	Updating generalization distances in the COFI Algorithm.	44
3.5	Training process in the COFI Algorithm.	46
3.6	Prediction process in the COFI Algorithm.	50
3.7	An example of execution of the prediction process in the COFI Algorithm.	51
3.8	An example of execution of the prediction process in the COFI Algorithm when test instance does not fall into any intervals.	53
3.9	An example concept description of the COFI Algorithm.	55
4.1	The concept description of the noise-free artificial dataset by the COFI Algorithm.	66
4.2	The concept description of the artificial dataset by the COFI Algorithm with one irrelevant feature.	67
4.3	Accuracy results of the COFI, the NN* and the NBC algorithms on domains with irrelevant attributes.	69
4.4	Memory requirement of the COFI, the NN* and the NBC algorithms on domains with irrelevant attributes.	70
4.5	Accuracy results of the the COFI, the NN* and the NBC algorithms on domains with unknown attribute values.	72
4.6	Memory requirement of the the COFI, the NN* and the NBC algorithms on domains with unknown attribute values.	73
A.1	Concept description of the iris dataset.	91
A.2	Concept description of the glass dataset.	92
A.3	Concept description of the glass dataset - continued.	93

A.4	Concept description of the glass dataset - continued.	94
A.5	Concept description of the horse-colic dataset.	95
A.6	Concept description of the horse-colic dataset - continued.	96
A.7	Concept description of the horse-colic dataset - continued.	97
A.8	Concept description of the horse-colic dataset - continued.	98
A.9	Concept description of the ionosphere dataset.	99
A.10	Concept description of the ionosphere dataset - continued. . . .	100
A.11	Concept description of the ionosphere dataset - continued. . . .	101
A.12	Concept description of the ionosphere dataset - continued. . . .	102
A.13	Concept description of the hungarian dataset.	103
A.14	Concept description of the hungarian dataset - continued. . . .	104
A.15	Concept description of the cleveland dataset.	105
A.16	Concept description of the cleveland dataset - continued. . . .	106
A.17	Concept description of the wine dataset.	107
A.18	Concept description of the wine dataset - continued.	108

List of Tables

4.1	An overview of the real-world datasets.	75
4.2	Required memory and execution time of training for the COFI Algorithm.	76
4.3	Accuracy results (%) of the algorithms for real-world datasets using leave-one-out evaluation technique.	77
4.4	Average memory requirements of the algorithms for real-world datasets.	78

List of Symbols and Abbreviations

AI	: Artificial Intelligence
COFI	: Classification by Overlapping Feature Intervals
c	: Index of a class
c_i	: Label of the i th class
c_t	: Time constant of the training process of the COFI algorithm
C4	: Decision tree algorithm
C4.5	: Decision tree algorithm
D_f	: Generalization distance for feature f in the COFI algorithm
D_{EH}	: Euclidean distance between example E and exemplar H
e	: An example
e_i	: i th example in the dataset
E	: An example
E_f	: f th feature value of the example E
E^n	: n -dimensional Euclidean space
f	: Index of a feature
f_i	: i th feature
g	: Generalization ratio
EACH	: Exemplar-Aided Constructor of Hyperrectangles
GA	: Genetic Algorithm
GA-CFP	: Hybrid CFP Algorithm
H	: Hyperrectangle
H_f	: f th feature value of the exemplar H
$H_{f,lower}$: Lower end of the range for the exemplar H for feature f
$H_{f,upper}$: Upper end of the range for the exemplar H for feature f
i	: Index of a training instance
IBL	: Instance-based learning
IB1	: Instance-based learning algorithm
IB2	: Instance-based learning algorithm
IB3	: Instance-based learning algorithm
IB4	: Instance-based learning algorithm
IB5	: Instance-based learning algorithm

ID3	: Decision tree algorithm
k	: Number of classes in the dataset
KA	: Knowledge Acquisition
log	: Logarithm in base 2
m	: Number of training instances
max_f	: Maximum value for the feature f
min_f	: Minimum value for the feature f
ML	: Machine Learning
n	: Number of features in the dataset
NBC	: Naive Bayesian Classifier
$ndi(f)$: Number of disjoint intervals for feature f
NGE	: Nested-Generalized Exemplars
NN	: Nearest Neighbor Algorithm
NN*	: Special case of the Nearest Neighbor Algorithm
$p(\mathbf{x} w_j)$: Conditional probability density function for \mathbf{x} conditioned on given w_j
$P(w_c)$: Prior probability of being class c for an instance
$P(w_c \mathbf{x})$: The posterior probability of an instance being class c given the observed feature value vector \mathbf{x}
$R(\alpha_i, \mathbf{x})$: Conditional risk
\mathbf{v}	: Vote vector
\mathbf{v}_f	: Vote vector of the feature f
$v_{f,c}$: The vote given to class c by feature f
\mathbf{x}	: Instance vector
\mathbf{x}_i	: Value vector of i th instance
$x_{i,f}$: Value of feature f of instance i
w_f	: Weight of feature f
w_H	: Weight of exemplar H
1R	: System whose input is training examples and output is 1-rule
Δ	: Weight adjustment rate of the CFP algorithm
$\lambda(\alpha_i, w_j)$: Loss incurred for taking action α_i when the state of nature is w_j

Chapter 1

Introduction

Learning refers to a wide spectrum of situations in which a learner increases his knowledge or skill in accomplishing certain tasks. The learner applies inferences to some material in order to construct an appropriate representation of some relevant aspect of reality. The process of constructing such a representation is a crucial step of in any form of learning [59].

One of the central insights of AI is that intelligence involves search, and that effective search is constrained by domain specific knowledge. This framework can be applied to problem solving, language understanding and learning from experience. One can even apply this search metaphor to machine learning as a field of scientific study [38]. In this framework, machine learning researchers are exploring a vast space of possible learning methods, searching for techniques with useful characteristics and looking for relations between these methods.

Machine learning is one of the oldest and most intriguing areas of artificial intelligence and cognitive science. It has developed especially in 80's, and has emerged as a subfield of AI that deals with techniques for improving the performance of a computational system. It is now distinguished from studies of human learning and from specific knowledge acquisition tools.

The aim of the machine learning and the statistical pattern recognition is to determine which category or class a given sample belongs to. Through an observation or measurement process, a set of values (usually numbers) which make up the observation vector serves as the input to a decision rule which

assigns the observed instance to one of the given classes [22]. In the statistical pattern recognition field, probabilistic approaches are used.

There are two kinds of major directions of AI research, symbolic and sub-symbolic models. The most active research area in recent years has continued to be symbolic empirical learning. This area is concerned with creating with and/or modifying general symbolic descriptions, whose structure is unknown *a priori*. Symbolic models are good in high-level reasoning, however they are weak in handling imprecise and uncertain knowledge. Subsymbolic models, such as neural networks and genetic algorithms, are powerful in lower-level reasoning especially imprecise classification and recognition problems. However, they are weak in high-level reasoning. Both of these models are important for understanding intelligent systems.

Learning from examples has been one of the primary paradigms of machine learning research since the early days of AI. Many researchers have observed and documented the fact that human problem solving performance improves with experience. In some domains, the principle source of expertise seems to be a memory for a large number of important examples. Attempts to build an intelligent (i.e., at the level of human) system have often faced the problem of memory for too many specific patterns. Researchers expect to solve this difficulty by building machines that can learn using limited resources. This reasoning has motivated many machine learning projects [48].

Inducing a general concept description from examples and counterexamples is one of the most widely studied method for symbolic learning. The goal is to develop a description of a concept from which all previous positive instances can be rederived while none of the previous negative instances can be rederived by the same process of rederivation of positive instances. Classification systems require only a minimal domain theory and they are based on the training instances to learn an appropriate classification function.

In this thesis, we propose a new symbolic model for concept learning, based on the representation of overlapping feature intervals. The Classification with Overlapping Feature Intervals algorithm is the particular implementation of this technique. In this new technique, overlapping concept descriptions are allowed, that is there may exist different classes for the same feature values. No specialization is done on the concept descriptions.

In the overlapping feature intervals technique, the basic unit of the representation is an *interval*. Each interval is represented by four parameters: *lower* and *upper bounds*, *the representativeness count*, which is the number of instances that the interval represents and *the associated class value of the interval*. The intervals are constructed through class dimensions for each feature. Initially, an interval is a point, that is, its lower and upper bounds are equal to initial feature values of the first instance for each feature. Then a point interval can be extended to a range interval such that its lower and upper bounds are not equal. This process is based on *generalization* through close interval heuristic [63]. During the training process, the set of values for each feature for each concept (class) is partitioned into segments (intervals) corresponding to concepts. That is, the concept description is the collection of intervals of each class dimension and the projection of these concept descriptions are learned by the COFI algorithm.

The ability to generalize from examples is widely recognized as an essential capability of any learning system. Generalization involves observing a set of training examples of some general concept, identifying the essential features common to these examples, then formulating a concept definition based on these common features. The generalization process can thus be viewed as a search through a vast space of possible concept definitions, in search for a correct definition of the concept to be learned. Because this space of possible concept definition is vast, the heart of the generalization problem lies in utilizing whatever training data, assumptions and knowledge available to constrain this search.

Generalization is the main process of the training phase of the COFI algorithm. The COFI algorithm does not use any specialization heuristic. In order to avoid overgeneralization of intervals, generalization is limited with a use of specified parameter. Therefore, the generalization of an interval to include a new training instance depends on the external variable, called *generalization ratio* and the maximum and the minimum feature value up to current example. By using this generalization ratio and these local maximum and minimum feature values a *generalization distance* is calculated. Whether a feature value is joined to an existing interval or it constructs a separate point interval is determined by this generalization distance. Small generalization ratios cause many number of small intervals to be constructed, whereas large generalization

ratios cause small number of large intervals.

The prediction process is simply a search for the intervals corresponding to the test instance's each feature value on the related class dimensions. If the test instance's current feature value falls within an interval in at least one class dimension, then a prediction can be made. Otherwise, no prediction is made and the final prediction is **UNDETERMINED**.

In the prediction phase of the COFI algorithm, local knowledge of each feature is important and this knowledge is maintained in a *vote vector*. For each feature a vote vector, whose elements represent the vote given to each class by that feature, is constructed. The vote is the *relative representativeness count*, which is the ratio of the representativeness count of the matched interval to the total number of instances which have the same class value as the test instance. Then, each feature's vote vector is summed for the final prediction. A voting is performed among the elements of this final vote vector, whose result is the class that receives the maximum vote in the final prediction of the COFI algorithm.

The COFI algorithm handles unknown attribute and class values in a straight forward way. Similar to human behavior, it just ignores these unknown attribute and class values. Most of the learning systems, usually overcome this problem by either filling in missing values with most probable value or a value determined by exploiting interrelationships among the values of different attributes or by looking at the probability distribution of known feature values [46].

Here, we have firstly given the introduction of machine learning and defined briefly the concept learning problem. Then an overview of the COFI algorithm has been presented. In Chapter 2, we will present some of the existing concept learning models. Then the detailed explanation about our new algorithm, COFI, will be given in Chapter 3. In Chapter 4, we evaluate the COFI algorithm by giving the complexity analysis and the results of empirical evaluation on artificial and real-world datasets. Finally, in Chapter 5, we will discuss the results of this study and conclude by giving general evaluation of the algorithm.

Chapter 2

Concept Learning Models

The field of machine learning (ML) is as difficult to define as its parent field artificial intelligence. One might describe it as the field of inquiry concerned with the processes by which intelligent systems improve their performance over time. However, such hard and fast definitions are no more successful at describing scientific disciplines than they are useful in characterizing everyday concepts. The best may be to describe the central tendency of the field, a tendency that may itself change as the field develops. For instance, machine learning shares with Artificial Intelligence (AI) a bias towards symbolic representations rather than numeric ones, although symbolic representation does not exclude the numeric one. Similarly, most machine learning research employs heuristic approaches to learning rather than algorithmic ones. These dimensions separate artificial and cognitive science from mainstream computer science and pattern recognition, and machine learning is much more closely associated with the former two areas than with the latter two.

Carbonell defines machine learning as follows [8]:

Perhaps the tenacity of ML researchers in light of the undisputed difficulty of their ultimate objectives, and in light of early disappointments, is best explained by the very nature of the learning process. The ability to learn, to adapt, to modify behavior is an inalienable component of human intelligence.

Despite the name machine learning, a significant fraction of the field has

always been concerned with modeling human behavior, starting with Feigenbaum's EPAM model of verbal learning [18]. There is also considerable interest in applied machine learning research, focusing on the automatic construction of knowledge-based systems. For instance, Michalski and Chilausky have worked on a knowledge base from examples [36]. Since learning is a central phenomenon in human cognition, the researchers evaluate machine learning methods in terms of their ability to explain human learning.

A number of different themes can be identified within the machine learning community, each corresponding to central goals of its parent field, artificial intelligence. For instance, many AI researchers are concerned with implementing knowledge-intensive systems, nevertheless those often take many man-years to construct. Machine learning may provide methods for automating this process, promising considerable savings in time and effort. Similarly, many AI researchers view artificial intelligence as a scientific discipline rather than an engineering one, and hope to formulate general principles of intelligent behavior that hold across a variety of domains. Since machine learning focuses on the acquisition of domain specific knowledge rather than the knowledge itself, it holds considerable potential for such general principles.

In the literature, two forms of learning can be distinguished: *Knowledge Acquisition (KA)* and *Refinement of Skills Through Experience* [59].

1. Knowledge Acquisition: Briefly, knowledge acquisition, like machine learning, describes techniques for increasing the functionality of a computer system. Knowledge acquisition focuses on the identification of knowledge for use in expert systems. Since this knowledge can be acquired in many ways, a wide variety of techniques have been studied. Knowledge acquisition and machine learning have been closely linked by their common application field, namely building up knowledge bases for knowledge-based systems. Learning and KA can be seen as two processes that construct a model of a task domain, including the systematic patterns of interaction of an agent situated in a task environment.

2. Refinement of Skills Through Experience: Learning in this perspective, consist of gradually correcting deviations between observed and desired behavior through repeated practice. This form of human learning covers

mental, motor and sensory processes. For example, learning a musical instrument is a good illustration of this process. Skill refinement is poorly understood, and few AI systems have attempted to simulate it to date.

As scientific databases continue to grow, the analysis of scientific data becomes an increasingly important application area for machine learning research. Much of the research on scientific data is directed towards predicting properties of a physical process. Such processes are often described in terms of a function defined over the attributes of the domain or a stochastic model. First, the best machine learning technique is selected for the problem domain. It has been argued that instead of producing a description of the problem domain in terms of logical rules, functional descriptions or a complex statistical model, it is possible to store a collection of memories (cases) and perform prediction by interpolating from them. It is shown that memory-based reasoning methods [47] are as effective as the more complex approaches, such as probabilistic approaches.

Many decision-making problems fall into the general category of classification. Empirical learning techniques for classification span roughly two categories: *statistical pattern recognition* [14, 22] and machine learning techniques for *induction of decision trees* [43] or *production rules*. Although a technique from either category is applicable to the same set of problems, the two categories of procedures can differ radically in their underlying models and final format of their solution. Both approaches to learning can be used to classify a sample pattern or example into a specific class.

Another distinction, *symbolic* and *subsymbolic learning models* concerns the level at which one represents instances and acquired knowledge. Many researchers in machine learning employ symbolic representations to describe both instances and rules [53]. In some cases, these involve complex logical or relational expressions, but a significant fraction of the work on inductive learning has employed attribute-value or feature representations of knowledge. However, inductive techniques also play a central role in subsymbolic learning paradigm, such as neural networks and evolutionary computing techniques [21, 40, 62]. In many cases, the inputs given to “subsymbolic” techniques are equivalent to the inputs provided to “inductive” methods.

There exist substantial and interesting differences among subsymbolic and

symbolic induction methods. Their learning algorithms and representations of knowledge differ in significant ways, and their inductive biases also appear to be quite different. However, all can be applied to the same class of induction tasks, and they can be compared to one another both experimentally and analytically. In the following sections, we will explain some of the basic machine learning and statistical pattern recognition techniques applied to concept learning tasks.

Concept Learning

The most widely studied method for symbolic learning is one of inducing a general concept description from examples and usually known counterexamples of the concept. The task is to build a concept description from which all previous positive instances can be rederived by universal instantiation while none of the previous negative instances (counterexample) can be rederived by the same process [8]. Until recently “learning” referred almost exclusively to classification mechanisms, focusing on programs that learn concept descriptions from a series of examples and counterexamples. While learning now extends to include many other topics and types of systems, classification is still an active field. Classification systems have only a minimal domain theory and rely almost exclusively on the training examples to learn an appropriate classification function.

Learning a concept usually means to learn its description, that is, a relation between the name of the concept and a given set of features that are used to describe instances. Several different representation techniques have been used to describe concepts for supervised learning tasks. One of the widely used representation technique is the *exemplar-based* representation. The representation of the concepts learned by the exemplar-based learning techniques stores only specific examples that are representativeness of other similar instances.

Previous implementations of the exemplar-based models usually extend the the nearest neighbor algorithm, in which some kind of similarity or distance metric is used for prediction. Therefore, the training complexity of such algorithms is proportional to the number of instances or objects stored. More recently, approaches using decision trees, connectionist architectures, representative instances, and hyperrectangles (exemplar-based learning) have appeared in the literature. These approaches construct concept descriptions by examining a series of examples, each of which is categorized as either a positive

example of the concept or a negative example (counterexample). That is, the main task is to construct a concept definition from the positive instances, which excludes the negative examples, by finding a relation between the name of the concept and a given set of features.

2.1 Exemplar-Based Learning

Exemplar-based learning was originally proposed as a model of human learning by Medin and Schaffer [35]. In the simplest form of exemplar-based learning, every example is stored in memory verbatim, with no change of representation. The set of examples that accumulate over time form category definitions; for example, the set of all chairs, that a person has seen, forms that person's definition of "chair". An example is normally defined as a vector of features, with values for each feature, plus a label which represents the category (class) of the example.

Exemplar-based learning is a widely used representation technique of the concept learning, in which the concept definition is constructed from the examples themselves, using the same representation language. In the exemplar-based learning, the examples are the concept. Little or no domain specific knowledge is required in exemplar-based learning.

In the literature, there are many different exemplar-based learning models. A hierarchical classification of exemplar-based learning algorithms is shown in Figure 2.1. All of these models share the property that they use verbatim examples as the basis of learning. For example, instance-based learning [4] retains examples in memory as points, and never changes them. An example concept description of this representation is shown in Figure 2.2. The main implementations of this technique are performed by Aha and Kibler, and the algorithms are known as IB1 through IB5 [2, 4]. The only decisions to be made are what points to store and how to measure similarity. Another example is the nested-generalized exemplars (NGE) of Salzberg. This model changes the point storage model of the instance-based learning and retains examples in the memory as axis-parallel hyperrectangles. Salzberg implemented NGE theory in a program called Exemplar-Aided Constructor of Hyperrectangles (EACH),

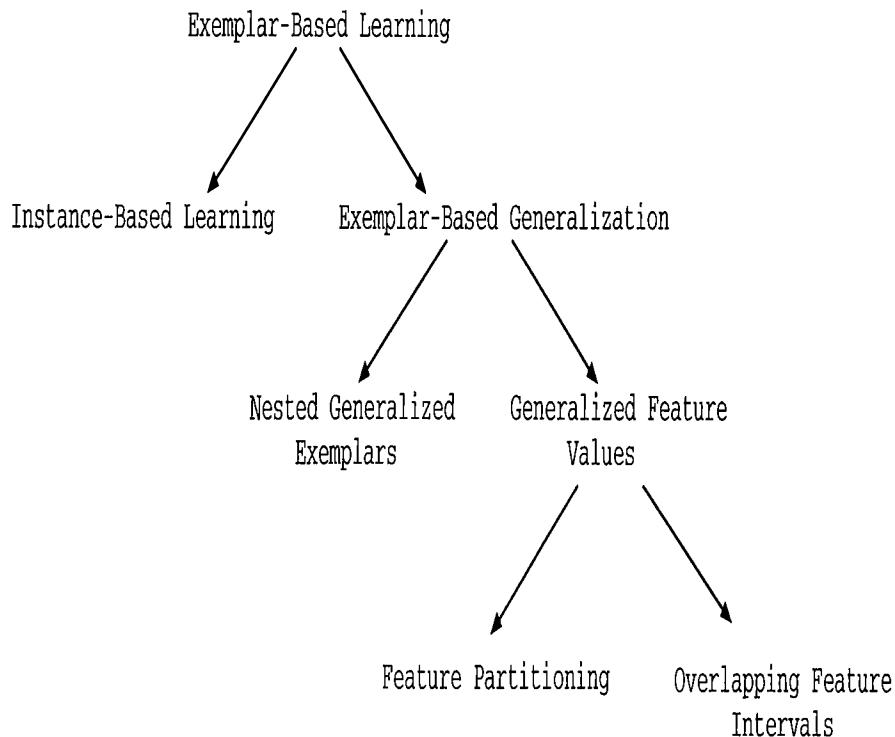


Figure 2.1. Classification of exemplar-based learning algorithms.

where numeric slots were used for feature values of exemplar [49]. An example of concept description is shown in Figure 2.3. In the feature partitioning techniques, examples are stored as partitions on the feature dimensions. One example of the implementation of feature partitioning is the Classification by Feature Partitioning (CFP) algorithm by Şirin and Güvenir [56]. An example concept description of this algorithm is presented in Figure 2.4. In the overlapping feature intervals technique, intervals are the representation of the concept descriptions. In this technique, the main property is to allow overlapped concept descriptions. The implementation of this technique is performed as the main purpose of this thesis and the algorithm is called Classification with Overlapping Feature Intervals (COFI).

2.1.1 Instance-Based Learning (IBL)

Instance-based learning algorithms use specific instances rather than precompiled abstractions during prediction tasks. These algorithms can also describe

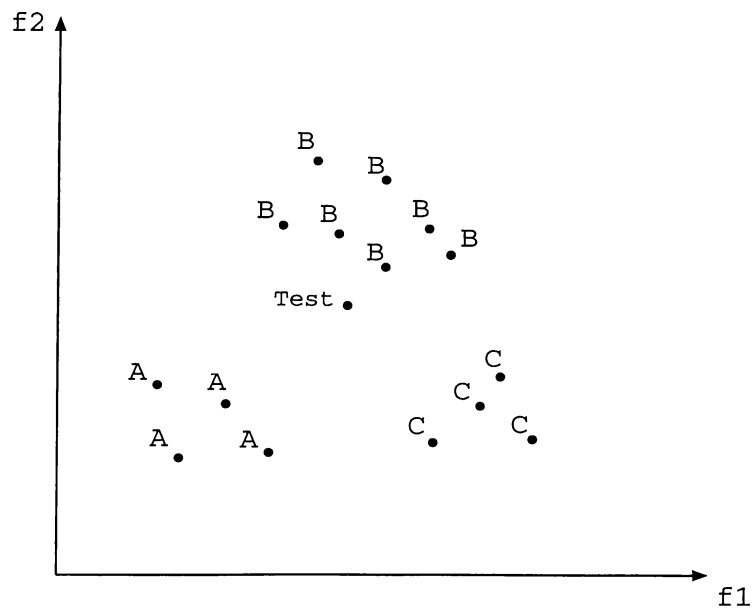


Figure 2.2. An example concept description in the IBL algorithms in a domain with two features.

probabilistic concepts because they use similarity functions to yield graded matches between instances.

IBL algorithms are derived from the nearest neighbor pattern classifier [10]. The primary output of the IBL algorithms is a *concept description* in the form of a set of examples. This is a function that maps instances to categories or classes.

In Figure 2.2, the representation of the concept description is shown. All examples are represented as points on the n -dimensional Euclidean space, where n is the number of features. In the figure, there are two features and three classes, namely A, B, and C. The test instance is marked as **Test**, and according to similarity function, this test instance will be classified as B.

An instance-based concept description includes a set of stored instances, called *exemplars*, and some information concerning their past performances during classification. This set of instances can change after each training instance is processed. However, IBL algorithms do not construct extensional concept descriptions. Instead, concept descriptions are determined by how the IBL algorithm's selected *similarity* and *classification* functions use the current set of saved instances. There are three components in the framework which

describes all IBL algorithms as defined by Aha and Kibler [4]:

1. **Similarity Function:** computes the *similarity* between two instances, similarities are real-valued.
2. **Classification Function:** receives the similarity function's results and the classification performance records of the instances in the concept description, and yields a classification for instances.
3. **Concept Description Updater:** maintains records on classification performance and decides which instance to be included in the concept description.

Five instance-based learning algorithms have been implemented: IB1, IB2, IB3, IB4 and IB5. IB1 is the simplest one and it uses the similarity function computed as

$$similarity(x, y) = -\sqrt{\sum_{f=1}^n (x_f - y_f)^2} \quad (2.1)$$

where x and y are the instances and n is the number of features that describes the instances. IB1 stores all the training instances. Therefore IB1 is not incremental, however, IB2 and IB3 are incremental algorithms. IB2's storage can be significantly smaller than IB1's, as it stores only the instances for which the prediction was wrong. IB3 is an extension of IB2. It employs a *significance test* to determine which instances are good classifiers and which ones are believed to be noisy. Once an example is determined to be noisy, it is removed from the description set. IB1 through IB3 algorithms assume that all attributes are equally relevant for describing concepts.

Extensions of these three algorithms [1, 2] are developed to remove some limitations which occur from some assumptions, such as, concepts are often assumed to

- be defined with respect to the same set of relevant attributes,
- be disjoint in instance space, and
- have uniform instance distributions.

IB4 [2] is an extension of IB3. It learns a separate set of attribute weights for each concept. These weights are then used in IB4's similarity function which is a Euclidean weighted-distance measure of the similarity of two instances. Multiple sets of weights are used because similarity is concept-dependent, the similarity of two instances varies depending on the target concept. IB4 decreases the affect of irrelevant attributes on classification decisions. It subsequently is more tolerant of irrelevant attributes.

The problem of novelty is defined as the problem of learning when novel attributes are used to help describe instances. IB4, like its predecessors, assumes that all the attributes used to describe training instances are known before training begins. However, in several learning tasks, the set of describing attributes is not known beforehand. IB5 [2], is an extension of IB4 that tolerates the introduction of novel attributes during training. To simulate this capability during training, IB4 simply assumes that the values for the (as yet) unused attribute are missing. During this time, IB4 fixates the expected relevance of the attribute for classifying instances. IB5 instead updates an attribute's weight only when its value is known for both of the instances involved in a classification attempt. IB5 can therefore learn the relevance of novel attributes more quickly than IB4. Theoretical analyses of instance-based learning algorithms can be found in [5].

Also noise-tolerant versions of instance-based algorithms are presented by Aha and Kibler in [3] in 1989. These learning algorithms are based on a form of significance testing, that identifies and eliminates noisy concept descriptions.

2.1.2 Nested-Generalized Exemplars (NGE)

Nested-generalized exemplar theory is a variation of exemplar-based learning. This theory is a model of a process whereby one observes a series of examples and becomes adept at understanding what those examples are examples of. Salzberg implemented NGE theory in a program called EACH (Exemplar-Aided Constructor of Hyperrectangles) [49], where numeric slots were used for feature values of exemplar. In EACH, the learner compares new examples to those it has seen before and finds the most similar example in memory. To determine the most similar hyperrectangle, a similarity metric, which is

inversely related to a distance metric (because it measures a kind of subjective distance), is used. This similarity metric is modified by the program during the learning process.

NGE theory makes several significant modifications to the exemplar-based model. It retains the notion that examples should be stored verbatim in memory, but once it stores them, it allows examples to be *generalized*. In NGE theory, generalizations take the form of hyperrectangles in a Euclidean n -space, where the space is defined by the variables measured for each example. The hyperrectangles may be nested one inside another to arbitrary depth, and inner rectangles serve as *exceptions* to surrounding rectangles [49]. The learner compares a new example to those it has seen before according to similarity metric. The term *exemplar* (or *hyperrectangle* in NGE) is used to denote an example stored in memory.

The system computes a match score between E and H , where E is a new data point and H is the hyperrectangle, by measuring the Euclidean distance between these two objects. Consider the simple case where H is a point, representing an individual example. The distance is determined by the usual distance function computed over every dimension in feature space,

$$D_{EH} = w_H \sqrt{\sum_{f=1}^n \left(w_f \frac{E_f - H_f}{\max_f - \min_f} \right)^2} \quad (2.2)$$

where w_H is the weight of the exemplar H , w_f is the weight of the feature f , E_f is the value of the f th feature on example E , H_f is the value of the f th feature on exemplar H , \max_f and \min_f are the minimum and maximum values of that feature, and n is the number of features recognizable on E .

If the exemplar H is not a point but a hyperrectangle, as being the case usually, the system finds the distance from E to the nearest face of H . There are obvious alternatives to this, such as using the center of H . The formula used above is changed because H_f , the value of the f th feature on H , is now a range instead of a point value. If we let $H_{f,lower}$ be the lower end of the range, and $H_{f,upper}$ be the upper end, then the equation becomes:

$$D_{EH} = w_H \sqrt{\sum_{f=1}^n \left(w_f \frac{d_f}{\max_f - \min_f} \right)^2} \quad (2.3)$$

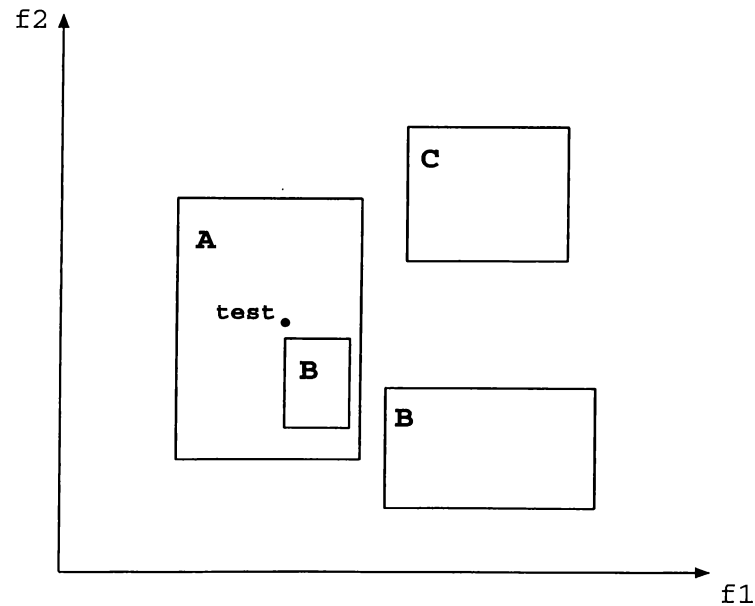


Figure 2.3. An example concept description of the EACH Algorithm in a domain with two features.

where

$$d_f = \begin{cases} E_f - H_{f,upper} & E_f > H_{f,upper} \\ H_{f,lower} - E_f & E_f < H_{f,lower} \\ 0 & otherwise \end{cases} \quad (2.4)$$

If a training instance E and exemplar H are of the same class, that is, a correct prediction has been made, the exemplar is generalized to include the new instance if it is not already contained in the exemplar. However, if the closest example has a different class then the algorithm modifies the weights of features so that the weights of the features that caused the wrong prediction is decreased.

In Figure 2.3, an example concept description of EACH algorithm is presented for two features f_1 and f_2 . Here, there are three classes, A, B and C, and their descriptions are rectangles (exemplars) as shown in Figure 2.3. It is seen in the figure that rectangle A contains another rectangle, B, in its region. B is an *exception* in the rectangle A. The NGE model allows exceptions to be stored quite easily inside hyperrectangles, and exceptions can be nested any number of levels. The test instance, that is marked as **test** in Figure 2.3, falls into the rectangle A, so the prediction will be the class value A for this test instance.

2.1.3 Classification By Feature Partitioning

The Classification by Feature Partitioning (CFP) algorithm [56, 57, 58, 61] is similar to the COFI algorithm, and it is the basis of this thesis. In the CFP algorithm, learning is done by storing the objects separately in each feature dimension. These stored objects are disjoint segments of feature values. For each segment, lower and upper bounds of the feature values, the associated class, and the number of instances it represents, representativeness count, are maintained. CFP learns segments of the set of possible values for each feature. An example is defined as a vector of feature values plus a label that represents the class of the related example.

Initially, a segment is a point on the representing feature dimension. A segment can be extended through generalization with other neighboring points of the same class in the same feature dimension. In order for a segment to be extended to cover a new point in feature f , the distance between the segment and the value of feature f must be less than a given generalization limit, which is defined separately for each feature.

The CFP algorithm pays attention to the disjointness of the segments. However, segments may have common boundaries on different features. In this case, weights of the features are used to determine the class value. The feature which has the maximum weight is chosen and this feature determines the class value. The training process in CFP has two steps: learning feature segments and learning feature weights.

The prediction in CFP is based on a vote taken among the predictions made by each feature separately. The effect of the prediction of a feature in the voting is proportional to the weight of that feature. All feature weights are initialized to 1 before the training process starts. The predicted class of a given instance is the one which receives the highest amount of votes among all feature predictions.

The first step in the training is to update the partitioning of each feature using the given training example. For each training example, the prediction of each feature is compared with the actual class of the example. If the prediction of a feature is correct, then the weight of that feature is multiplied by $(1 + \Delta)$ otherwise, it is multiplied by $(1 - \Delta)$, where Δ is called the global feature

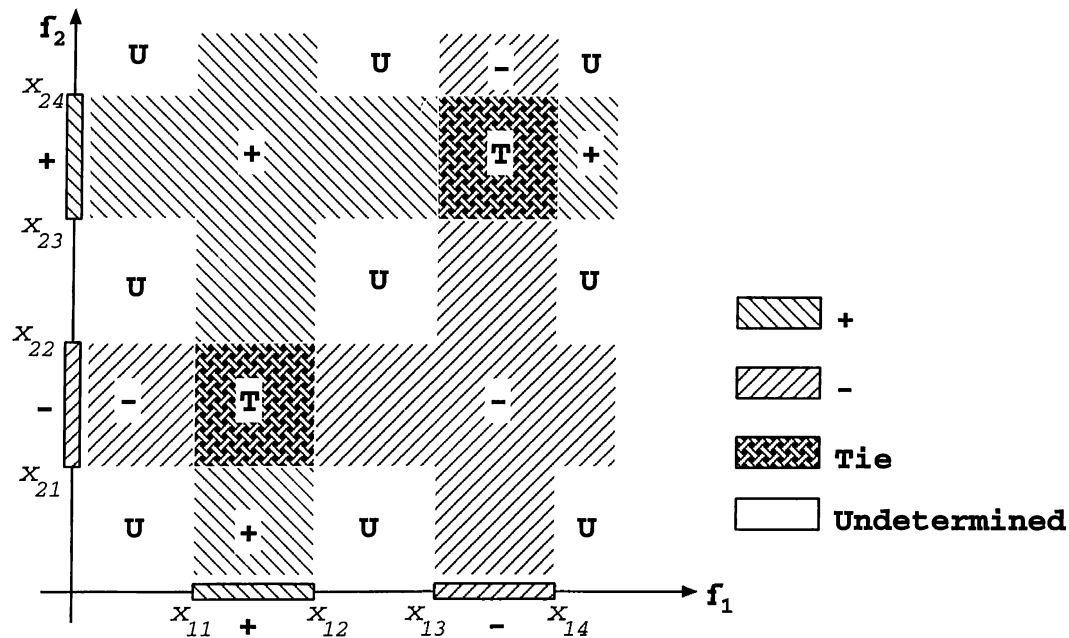


Figure 2.4. An example concept description of the CFP Algorithm in a domain with two features.

weight adjustment rate.

Figure 2.4 is presented here in order to illustrate the form of the resulting concept descriptions learned by the CFP algorithm with two features f_1 and f_2 . Assume that there are two classes, positive (+) and negative (-), and their boundaries are shown in Figure 2.4. If a test instance falls into a region that marked with +, then the prediction will be the positive class for this instance in Figure 2.4. Similarly, if it falls into a region that marked with -, then the prediction will be the negative class. However, if the test instance falls into a region with marked with U, then no prediction will be made for this instance. Finally, if the test instance falls into a region that marked with T then the prediction will be made according to the weights of the features, that is, the feature which has the maximum weight makes its segment's class value as the final prediction.

A noise tolerant version of the CFP algorithm has also been developed. In this extension, the CFP algorithm removes the segments that are believed to be introduced by noisy instances. A new parameter, called confidence threshold (or level) is introduced to control the process of removing the partitions from the concept description. The confidence threshold and observed frequency of

the classes are used together to decide whether a partition is noisy.

Also a hybrid system, called GA-CFP, which combines a genetic algorithm with the CFP algorithm has been developed [24]. The genetic algorithm is used to determine a very good set of domain dependent parameters ¹ of the CFP, even when trained with a small partition of the data set. An algorithm that hybridizes the classification power of the feature partitioning CFP algorithm with the search and optimization power of the genetic algorithm is presented. The resulting algorithm GA-CFP requires more computational capabilities than the CFP algorithm, but achieves improved classification performance in reasonable time. The complexity analysis of the CFP algorithm is presented in [25].

A limitation of the CFP algorithm can be seen in the following example in Figure 2.5. Here, the domain has only one dimension. In Figure 2.5A, the order of the training dataset is given. The first instance has the class value c_1 and this instance constructs a point segment at the feature value x_1 on the related feature dimension initially, as shown in Figure 2.5A.a. Then second example is processed, it has also the class value c_1 and its feature value is x_2 . Here, we assumed that the generalization distance is greater than the difference between x_1 and x_2 . Therefore, a range segment is constructed on the feature dimension and its lower bound is x_1 and upper bound is x_2 as shown in Figure 2.5A.b. Then, let the next five instances belong to another class c_2 , and their related feature values are between x_1 and x_2 . In this case, the big segment in Figure 2.5A.b has subpartitioned into six segments for class c_1 and no segment can be constructed for the second class c_2 as shown in Figure 2.5A.c.

However, if the last five instances were processed before the first two instances in the previous example, then the segments would have been constructed as shown in Figure 2.5B. Firstly, a range segment is constructed for the class c_2 as shown in Figure 2.5B.a, and then two point segments are constructed for the last two instances as in Figure 2.5B.b. The concept descriptions (segments) for A and B cases are very different from each other in Figure 2.5, although the same training instances were processed. It is seen that the order of the instances is very important and it affects the resulting concept description considerably.

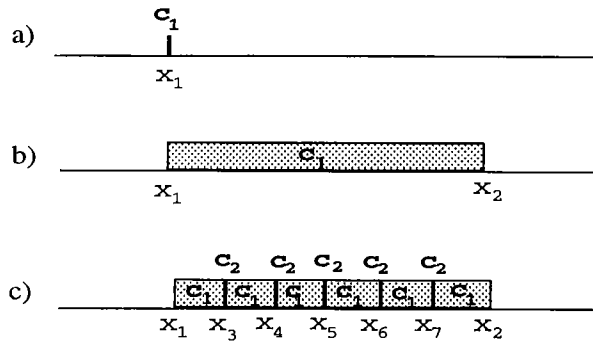
In order to overcome this limitation, we need to store the instances of

¹ Δ and D_f for each feature

A)

Order of the training dataset

- $i_{1,1} = x_1$
- $i_{2,1} = x_2$
- $i_{3,1} = x_3$
- $i_{4,1} = x_4$
- $i_{5,1} = x_5$
- $i_{6,1} = x_6$
- $i_{7,1} = x_7$



B)

Order of the training dataset

- $i_{1,1} = x_3$
- $i_{2,1} = x_4$
- $i_{3,1} = x_5$
- $i_{4,1} = x_6$
- $i_{5,1} = x_7$
- $i_{6,1} = x_1$
- $i_{7,1} = x_2$

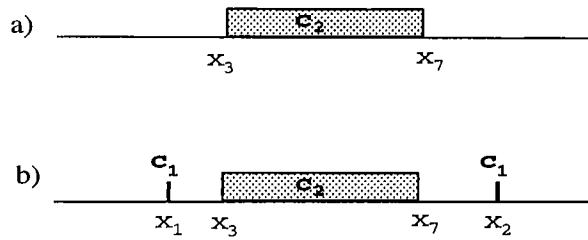


Figure 2.5. Constructing segments in CFP by changing the order of the training dataset.

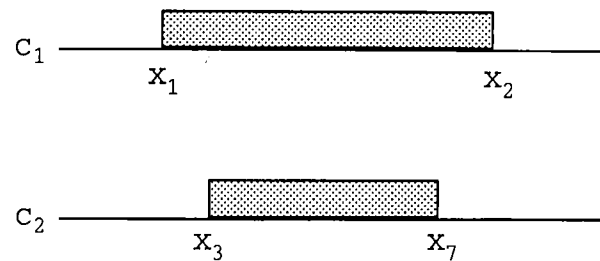


Figure 2.6. Constructing intervals in the COFI Algorithm with using same dataset as used in Figure 2.5 .

different classes separately in a feature dimension independent from each other. The COFI algorithm solves this problem by storing examples as overlapping intervals for each class separately. The concept description learned by the COFI algorithm from the same set of training instances is shown in Figure 2.6 independent from the order of the examples, when the generalization ratio is chosen properly.

2.2 Decision Tree Techniques

Another approach to inductive learning is the one which involves the construction of decision trees. Here the concept representation is done by using tree structure.

A decision tree can be used to classify a case by starting at the root of the tree and moving through it until a leaf is encountered. At each non-leaf decision node, the outcome of the case for the test at the node is determined and attention shifts to the root of the subtree corresponding to this outcome. When this process finally leads to a leaf, the class of the case is predicted to be that record at the leaf.

2.2.1 Decision Trees

This method was developed initially by Hunt, Marin and Stone in 1966 [27], and later modified by Quinlan (1979, 1983), who applied his ID3 algorithm to deterministic domains such as chess and games [41, 42]. Quinlan's later

research has focused on induction on domains that are uncertain and noisy rather than deterministic. His approach is to synthesize decision trees that has been used in a variety of systems, and he has described his system ID3, the details can be found in [41, 42, 43]. A more extended version of ID3 is C4.5 [46] which can convert a decision tree to a rule base.

The overall approach employed by ID3 and C4.5 is to choose the attribute that best divides the examples into classes and then partition the data according to the values of that attribute. This process is recursively applied to each partitioned subset, with the procedure terminating when all examples in the current subset have the same class. The result of this process is represented as a tree in which each node specifies an attribute and each branch emanating from a node specifies the set of possible values of that attribute. Terminal nodes (leaves) of the tree correspond to sets of examples with the same class or to cases in which no more attributes are available.

The recursive partitioning method of constructing decision trees will continue to subdivide the set of training cases until each subset in the partition contains cases of a single class, or until no tests offers any improvement. The result is often a very complex tree that “overfits the data” by inferring more structure than is justified by the training cases. A decision tree is not usually simplified by deleting the whole tree in favor of a leaf. Instead, the idea is to remove parts of the tree that do not contribute to classification accuracy on unseen cases, producing something less complex and thus more comprehensible. This process is known as the *pruning*. There are basically two ways in which the recursive partitioning method can be modified to produce simpler trees: deciding not to divide a set of training cases any further, or removing retrospectively some of the structure built up by recursive partitioning [46].

The former approach, sometimes called *stopping* or *prepruning*, has the attraction that time is not wasted assembling structure that is not used in the final simplified tree. The typical approach is to look at the best way of splitting a subset and to assess the split from the point of view of statistical significance, information gain, error reduction, or whatever. If this assessment falls below some threshold then the division is rejected.

The difference between this approach and the other learning methods, including the COFI algorithm, is that the performance of these systems does not

depend critically on any small part of the model, whereas decision trees are much more susceptible to small changes.

2.2.2 1R System

In this section, the specific kind of rules, called “1-rules”, are examined. These are the rules that classify an object on the basis of a single attribute that is, they are 1-level decision trees. This system is defined by Holte, in 1993 [26].

1R is a system whose input is a set of training examples and whose output is a 1-rule. 1R can be treated as a special case of the classification with overlapping feature intervals technique. The COFI algorithm uses all of the feature information for a final prediction. However, in 1R, one of the features is chosen to make the final prediction. 1R system treats all numerically valued attributes as continuous and uses a straight forward method to divide range of values into several disjoint intervals. Similar to the COFI algorithm, 1R handles the unknown attribute values by ignoring them. 1R makes each interval “pure”, that is, intervals contain examples that are all of the same class. 1R requires all intervals, except the rightmost, to contain a more than a predefined number of examples.

During the training phase, the construction of the intervals in the 1R rule is done. After the training phase, one of the feature’s concept description is chosen as a rule.

Holte shows that simple rules such as 1R are as accurate as more complex rules such as C4. Given a dataset, 1R generates its output, a 1-rule, in two steps. First, it constructs a relatively small set of candidate rules (one for each attribute), and then it selects the one that makes the smallest error on the training dataset. This two steps pattern is similar to the training process of many learning systems.

In [26], Holte used sixteen datasets to compare 1R and C4, and fourteen of the datasets were selected from the collection of datasets distributed by the machine learning group at the University of California at Irvine. The main result of comparing 1R and C4 is an insight into the tradeoff between simplicity and accuracy. 1R rules are only a little less accurate (3.1 percentage points)

than C4's pruned decision trees on almost all of the datasets. Decision trees formed by C4 are considerably larger in size than 1-rules.

The fact that, on many datasets, 1-rules are almost as accurate as more complex rules has important implications for machine learning research and applications. The first implication is that 1R can be used to predict the accuracy of the rules produced by more sophisticated machine learning systems. A more important implication is that simple-rule learning systems are often a viable alternative to systems that learn more complex rules.

2.3 Statistical Concept Learning

The purpose of the statistical concept learning (or statistical pattern recognition) is to determine to which category or class a given sample belongs to as for the machine learning. It is felt that the decision-making processes of a human being are somewhat related to the recognition of patterns; for example the next move in chess game is based upon the present pattern on the board, and buying or selling stocks is decided by a complex pattern of information. The goal of the pattern recognition is to clarify these complicated mechanisms of decision-making processes and to automate these functions using computers.

Pattern recognition, or decision-making in a broader sense, may be considered as a problem of estimating density functions in a high-dimension space and dividing the space into the regions of categories or classes. Because of this view, mathematical statistics forms the foundations of the subject.

Several classical pattern recognition methods have been presented in the literature [13, 14, 22, 62]. Some of them are parametric, that is, they are based on assumed mathematical forms for either the density functions or the discriminant functions.

The Bayesian approach to classification estimates the (posterior) probability that an instance belongs to a class, given the observed attribute values for the instance. When making a categorical rather than probabilistic classification, the class with the highest estimated posterior probability is selected.

The posterior probability, $P(w_c|\mathbf{x})$ of an instance being class c , given the observed attribute value vector \mathbf{x} , may be found in terms of the prior probability of an instance being class c , $P(w_c)$; the probability of an instance of class c having the observed attribute values, $P(\mathbf{x})$. In the Naive Bayesian approach, the likelihoods of the observed attribute values are assumed to be mutually independent [14, 22, 60].

Naive Bayesian classifier is one of the common parametric classifiers. When no parametric structure can be assumed for the density functions, nonparametric techniques, for instance nearest neighbor method, must be used for classifications. Here we will explain Bayes Independence (Naive Bayesian Classifier) and Nearest Neighbor methods because of their similarities to the COFI algorithm.

Nearest neighbor method is one of the simplest methods conceptually, and is commonly cited as a basis of comparison with other methods. It is often used in case-based reasoning [51].

Bayes rule is the optimal presentation of minimum error classification. All classification methods can be viewed as approximations to Bayes optimal classifiers. This theory is the fundamental statistical approach to the problem of pattern classification. This approach is based on the assumption that the decision problem is posed in probabilistic terms, and that all of the relevant probability values are known.

Both NBC and NN algorithms are non-incremental, that is, they take and process all the training instances at once. On the other hand, the COFI algorithm has an incremental structure. The COFI algorithm process each example separately and when new instances are fetched, it updates its concept description, that is, it does not have to know all the feature values at once. On the other hand, Naive Bayesian Classifier (NBC), and the version of nearest neighbor algorithm (NN*), similar to the COFI algorithm, process each feature separately. Therefore, we will compare the COFI algorithm with the NBC and the NN* algorithms.

2.3.1 Naive Bayesian Classifier (NBC)

The computation of the *a posteriori* probabilities $P(w_c|\mathbf{x})$ lies at the heart of the Bayesian classification. Here w_c is the class and \mathbf{x} is the feature vector of the instance to be classified. Bayes rule allows us to compute the probabilities $P(w_c)$ and the class conditional densities $p(\mathbf{x}|w_c)$, but since these quantities are unknown, the best is to compute $P(w_c|\mathbf{x})$ using all of the information at disposal.

Let $\Omega = \{w_1, w_2, \dots, w_k\}$ be the finite set of s states of nature and $A = \{\alpha_1, \alpha_2, \dots, \alpha_a\}$ be the finite set of a possible actions. Let $\lambda(\alpha_i, w_j)$ be the loss incurred for taking action α_i when the state of nature is w_j . Let the feature vector \mathbf{x} be a vector-valued random variable, and let $p(\mathbf{x}|w_j)$ be the state-conditional probability density function for \mathbf{x} , that is, the probability density function for \mathbf{x} conditioned on w_j being the state of nature. Finally, let $P(w_j)$ be the *a priori* probability that nature is in the state $P(w_j)$. That is, $P(w_j)$ is the proportion of all instances of class j in the training set. Then the *a posteriori* probability $P(w_j|\mathbf{x})$ can be computed from $p(\mathbf{x}|w_j)$ by Bayes rule:

$$P(w_j|\mathbf{x}) = \frac{p(\mathbf{x}|w_j)P(w_j)}{p(\mathbf{x})} \quad (2.5)$$

where

$$p(\mathbf{x}) = \sum_{j=1}^s p(\mathbf{x}|w_j)P(w_j). \quad (2.6)$$

Since $P(w_j|\mathbf{x})$ is the probability that the true state of nature is w_j , the expected loss associated with taking action α_i is merely

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^s \lambda(\alpha_i|w_j)P(w_j|\mathbf{x}). \quad (2.7)$$

In decision theoretic terminology, an expected loss is called *risk*, and $R(\alpha_i|\mathbf{x})$ is known as the *conditional risk*. Whenever we encounter a particular observation \mathbf{x} , we can minimize our expected loss by selecting the action that minimizes the conditional risk. Now, our problem is to find a Bayes decision rule against $P(w_j)$ such that minimizes the overall risk. A decision rule is a function $\alpha(\mathbf{x})$ that tells us which action to take for every possible observation. That is, for

```

nbc(test example e):
begin
  prediction = 0
  foreach class c
    /*class_count[c]:# of examples that have the class value c */
    g[c] = class_count[c] / (no of training examples)

    foreach feature f
      g[c] = g[c] * probability(e, f, c)

    if g[c] > g[prediction] then
      prediction = c

  return(prediction)
end.

```

Figure 2.7. The NBC Algorithm.

every \mathbf{x} , the decision function $\alpha(\mathbf{x})$ assumes one of the a values $\alpha_1, \alpha_2, \dots, \alpha_a$. The overall risk R is the expected loss associated with a given decision rule. To minimize the overall risk, we compute the conditional risk for $i = 1, \dots, a$ and select the action α_i for which $R(\alpha_i|\mathbf{x})$ is minimum. The resulting minimum overall risk is called the *Bayes risk* and is the best performance that can be achieved.

$$R(\alpha_i|\mathbf{x}) = \sum_{j=1}^s \lambda(\alpha_i|w_j)P(w_j|\mathbf{x}) \quad (2.8)$$

The probability of error is the key parameter in pattern recognition. The error due to the Bayes classifier gives the smallest error from given distributions.

There are many different ways to represent pattern classifiers. One way is in terms of a set of discriminant functions $g_i(x)$, $i = 1, \dots, k$ where k is the number of classes. The classifier is set to assign a feature vector \mathbf{x} to class w_i if

$$g_i(x) > g_j(x) \quad \text{for all } i \neq j. \quad (2.9)$$

Thus, the classifier is viewed as a machine that computes discriminant functions

and selects the category whose discriminant function has the largest value.

For the general case we can let $g_i(x) = -R(\alpha|x)$ where $-R(\alpha|x)$ is the conditional risk, since the maximum discriminant function will then correspond to the minimum conditional risk. For the minimum-error-rate case, things can be simplified by taking $g_i(x) = P(w_i|x)$, so that the maximum discriminant function corresponds to the maximum *a posteriori* probability. The algorithm for the computing these *a posteriori* probabilities is shown in Figure 2.8.

The choice of discriminant functions is not unique. More generally, if every $g_i(x)$ is replaced by $f(g_i(x))$, where f is a monotonically increasing function, the resulting classification is unchanged. This observation can lead to significant analytical and computational simplifications. In particular, for minimum-error-rate classification, any of the following choices gives identical classification results, but some can be much simpler to understand or to compute than others:

$$g_i(\mathbf{x}) = P(w_i, \mathbf{x}) \quad (2.10)$$

$$g_i(\mathbf{x}) = \frac{P(\mathbf{x}|w_i)P(w_i)}{\sum_{j=1}^c P(\mathbf{x}|w_j)P(w_j)} \quad (2.11)$$

$$g_i(\mathbf{x}) = P(\mathbf{x}|w_i)P(w_i) \quad (2.12)$$

$$g_i(\mathbf{x}) = \log P(\mathbf{x}|w_i) + \log P(w_i) \quad (2.13)$$

Even though the discriminant functions can be written in a variety of forms, the decision rules are equivalent. The effect of any decision rule is to divide the feature space into k decision regions, R_1, \dots, R_k . If $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $i \neq j$, then \mathbf{x} is in R_i and the decision rule calls for us to assign \mathbf{x} to w_i . The regions are separated by decision boundaries, surfaces in feature space where ties occur among the largest discriminant functions. If R_i and R_j are contiguous, the equation for the decision boundary separating them is

$$g_i(\mathbf{x}) = g_j(\mathbf{x}). \quad (2.14)$$

While this equation may appear to take different forms depending on the forms chosen for the discriminant functions, the decision boundaries are, of course,

```

probability(e, f, c):
begin
  foreach train example train_e
    if train_e's class = c then
      if train_e's feature value = f then
        equalc++
      elseif train_e's feature value < f then
        if train_e's feature value = largest_of_smaller then
          lsc++
        elseif train_e's feature value > largest_of_smaller then
          largest_of_smaller = train_e's feature value
          lsc = 1
      else
        if train_e's feature value = smallest_of_larger then
          slc++
        elseif train_e's feature value < smallest_of_larger then
          smallest_of_larger = train_e's feature value
          slc = 1

  if equalc = 0 then
    if largest_of_smaller == -INFINITY then
      density = slc / (smallest_of_larger - x) / class_count[c]
    elseif smallest_of_larger = INFINITY then
      density = lsc / (x - largest_of_smaller) / class_count[c]
    else
      difference = smallest_of_larger - largest_of_smaller
      density = (slc+lsc) / 2 / difference / class_count[c]

  elseif largest_of_smaller= -INFINITY then
    density = equalc / (smallest_of_larger - x) / class_count[c]
  elseif smallest_of_larger= INFINITY then
    density = equalc / (x - largest_of_smaller) / class_count[c]
  else
    difference = smallest_of_larger - largest_of_smaller
    density = equalc / difference * 2 / class_count[c]

  return (density)
end.

```

Figure 2.8. Computing the *a posteriori* probabilities in the NBC Algorithm.

the same. For points on the decision boundary, the classification is not uniquely defined. For a Bayes classifier, the conditional risk associated with either decision is the same, and it does not matter how ties are broken. The algorithm of the NBC is given in Figure 2.7.

NBC processes each feature separately as the COFI algorithm does, that is, probabilities are independent from each other:

$$P(\mathbf{x}|w_c) = \prod_{f=1}^n P(x_f|w_c). \quad (2.15)$$

For this reason the comparison of these methods is useful and the results of this comparison is given in Chapter 4.

2.3.2 The Nearest Neighbor Classification

Let $X^n = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ be a set of n labeled samples, and let $\mathbf{x}'_n \in X^n$ be the sample nearest to \mathbf{x} . Then the nearest neighbor rule for classifying \mathbf{x} is to assign it the label associated with \mathbf{x}'_n . The nearest neighbor rule is a suboptimal procedure, its use usually will lead to an error rate greater than the minimum possible Bayes rate.

In this thesis, a special case of the nearest neighbor (NN) classification method is used and this case is represented by NN*. In this special case each feature is processed independently as processed in the NBC and the COFI algorithms. We have used NN* in our comparisons, because the COFI, the NBC and the NN* algorithms share a common property of processing each feature separately and applying a voting mechanism to make the final prediction. In NN* all the examples are stored in memory as verbatim. In order to make predictions, the distance between test instances and training examples should be computed.

In the NN* algorithm, for each feature, the nearest training instance to the test instance is determined independently. Here the distance between a test instance x and a training instance y on feature f is defined as

$$distance_f(x, y) = x_f - y_f. \quad (2.16)$$

```
nn(test instance e):
begin
  foreach class c
    vote[c] = 0

  foreach feature f
    distance to nearest = infinite
    foreach train example train_e
      distance = distance(e, train_e, f)
      if distance < distance to nearest then
        distance to nearest = distance
        foreach class c
          nearest_count[c] = 0
          nearest_count[train_e's class value] = 1
        elseif (distance = distance to nearest) then
          nearest_count[train_e's class value] ++;
    foreach class value c
      vote[c] = vote[c] + nearest_count[c]

  prediction = 0
  for each class c
    if vote[c] > vote[prediction] then
      prediction = c

  return (prediction)
end.
```

Figure 2.9. The NN* Algorithm.

Here x_f is the f th feature value of the test instance x and y_f is the f th feature value of the training instance y .

All examples are processed once for each feature, that is, the distance between all examples' first feature value and the test instance's first feature value is computed separately. The training example which has the minimum distance (nearest instance in terms of the first feature) gives its class value as prediction for the corresponding feature, in our example this is the first feature. This process is repeated for all features. At the end, after all features make their predictions, the class value which is predicted mostly is the final prediction. The algorithm of this process is given in Figure 2.9. According to this distance metric, all feature values should be searched to find the minimum distance. Since the feature values are not stored in an ordered form in the real-world datasets, a sequential search gives the prediction for each feature. Therefore, the prediction complexity of this algorithm for one instance is $O(nm)$ where n is the number of features, and m is the number of instances.

In Chapter 4, we will compare these statistical concept learning methods, the NBC and the NN* algorithms with the COFI algorithm in terms of accuracy and the space complexity by using some artificial and real-world datasets.

Chapter 3

Classification with Overlapping Feature Intervals

In this chapter the detailed explanation of the Classification with Overlapping Feature Intervals (COFI) will be presented. The COFI algorithm is an exemplar-based concept learning algorithm. Learned concepts are represented as intervals on the class dimensions for each feature. Making predictions depends on the search over these intervals and voting among the predictions that all the features made.

The characteristics of the COFI algorithm will be presented firstly. The characteristic features will be compared with other learning techniques. Then, training and testing in the COFI algorithm will be presented and discussed, and the execution of the algorithm will be explained through examples.

3.1 Characteristics of the COFI Algorithm

Obviously, a learning system should have some characteristics to perform the learning task. However, these characteristics differ from one technique to another one. Here, the general characteristics of the COFI algorithm are presented and then compared to some other learning methods.

3.1.1 Knowledge Representation Schemes

One of the most useful and interesting dimension in classifying machine learning techniques is the way they represent the knowledge they acquire. Many systems acquire *rules* during the learning process.

Typically such systems will try to generalize the antecedents of rules, so that those rules apply to as large a number of situations as possible. Another way to represent what is learned is with decision trees as in the ID3 and C4.5 algorithms [46].

Usually exemplar-based learning models produce neither rules nor decision trees. Instead, they create a memory space filled with exemplars, many of which represent generalizations and some of which represent individual examples from the system's experience. In the IBL algorithms, individual instances are stored as shown in Figure 2.2. In the NGE algorithm, the exemplars are hyperrectangles, i.e., rectangular solids in E^n [49] as shown in Figure 2.3.

In the COFI algorithm, the knowledge representation is similar to the NGE method. In this method no domain theory is used as in exemplar-based learning algorithms. Learned concepts are stored in memory and called *intervals*. These intervals are stored as a list for each feature and for each class. Each element (node) of this list contains upper and lower bounds of the interval, representativeness count, that is the number of examples that interval represents, and the associated class value of the interval. The number of intervals depends on the example dataset, generalization distances and the coming order of the examples. At the worst case, if all examples are nominal and have different feature values, that is, they are unique, then the number of intervals is equal to mn where m is the number of instances, and n is the number of features. However, if example feature values are linear and generalization distances are greater than zero, there will be much less number of intervals. The learned intervals are presented in the Appendix for each real-world datasets that are used for empirical evaluation of the COFI algorithm.

The conversion of intervals to the if-then rules is also possible for the COFI algorithm. The explanation of this process is given in this chapter, in Subsection 3.2.4.

3.1.2 Supervised Learning

Historically, the most important learning paradigm has been that of *supervised learning*. In this framework, the learner is asked to associate pairs of items. When later presented with just the first item of a pair, the learner is supposed to recall the second. This paradigm has been used in pattern classification, concept acquisition tasks, learning from examples, system identification and associative memory. For example, in pattern classification or concept acquisition, the first item is an instance of some pattern or concept and the second item is the name of the concept. In system identification, the learner must reproduce the input-output behavior of some unknown system. Here, the first item of each pair is an input and the second item is the corresponding output.

Any prediction problem can be cast in the supervised learning paradigm by taking the first item to be the data based on which a prediction must be made, and the second item to be the actual outcome what the prediction should have been.

On the other hand, in *unsupervised learning* techniques, the task is to estimate the class distributions successively, assuming that we do not know the true distributions from which the incoming samples are taken. This is also termed *learning without a teacher*. Because of the additional ambiguity, the computation of unsupervised estimation becomes more complex. However, the development of this kind of techniques is motivated by the hope that the machine may improve the performance without any outside supervision after initial learning in a supervised mode.

In the COFI algorithm, the supervised learning paradigm is followed. Here the first item is the attribute values of an instance and the second item is the class of that instance.

3.1.3 Incremental Learning

Strategies employed by most systems fall into one of two main categories as Quinlan has pointed out. The machine learning systems using *incremental learning strategies* attempt to improve an internal model with each example

they process. In *batch (non-incremental) learning strategies*, a program must see all of the input examples before beginning to build a model of the domain.

Researchers who explore the incremental approach are typically concerned with developing plausible models of human learning, with agents that must interact with a dynamic environment, or with the efficiency of the learning mechanisms. In contrast, those who employ non-incremental learning methods are typically concerned with automating the process of knowledge acquisition for expert systems.

A non-incremental learning strategy usually assumes random access to the examples in the training set. A learning system which follows this strategy search for patterns and regularities in the training set in order to formulate decision trees or rules. They may examine and re-examine the training set many times before settling on a good model. The advantage of this approach is that it is not sensitive to the order of the training examples.

Despite the differences in motivation, researchers in both paradigms have much to learn from each other. Incremental and non-incremental systems often use the same basic learning operators and produce similar results. In many cases, one can create incremental variations of non-incremental algorithms. Presumably, many incremental learning methods also have non-incremental counterparts.

The COFI algorithm employs the incremental learning strategy. The learning task is performed in a dynamic environment. In the COFI algorithm, system represents the concept definition by means of the intervals. The construction of intervals depends on the generalization distances. Since generalization distance is determined according to current maximum and minimum feature values, the order of the instances affects the boundaries of the intervals. That is, generalization distance values depend on the order of the data. If generalization distances had static values then the order of the data could not be important and the COFI algorithm could not behave in an incremental manner. When a new training example is fetched, then the generalization distance and the related interval are updated. This is very similar to the human learning of ranges. One example of this incremental structure is illustrated by the following example where the dataset contains three instances, and the domain has a single feature. Let the training instance's first feature values be

Order of training instances (A)

$$i_{1,1} = 1 \quad c = 1$$

$$i_{2,1} = 4 \quad c = 1$$

$$i_{3,1} = 10 \quad c = 1$$

generalization ratio $g = 0.5$

a) execution for i_1

$$f_1 \quad \begin{array}{c} c_1 \\ | \\ \text{-----} \\ 1 \end{array} \quad \langle [1, 1], 1, c_1 \rangle \quad D_1 = 0.0$$

b) execution for i_2

$$f_1 \quad \begin{array}{c} c_1 \quad c_1 \\ | \quad | \\ \text{-----} \\ 1 \quad 4 \end{array} \quad \begin{array}{l} \langle [1, 1], 1, c_1 \rangle \quad D_1 = 1.5 \\ \langle [4, 4], 1, c_1 \rangle \end{array}$$

c) execution for i_3

$$f_1 \quad \begin{array}{c} c_1 \quad c_1 \quad c_1 \\ | \quad | \quad | \\ \text{-----} \\ 1 \quad 4 \quad 10 \end{array} \quad \begin{array}{l} \langle [1, 1], 1, c_1 \rangle \quad D_1 = 4.5 \\ \langle [4, 4], 1, c_1 \rangle \\ \langle [10, 10], 1, c_1 \rangle \end{array}$$

Figure 3.1. An example of construction of the intervals in the COFI Algorithm.

Order of training instances (B)

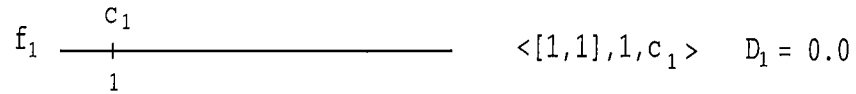
$$i_{1,1} = 1 \quad c = 1$$

$$i_{2,1} = 10 \quad c = 1$$

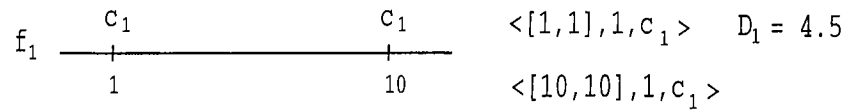
$$i_{3,1} = 4 \quad c = 1$$

generalization ratio $g = 0.5$

a) execution for i_1



b) execution for i_2



c) execution for i_3

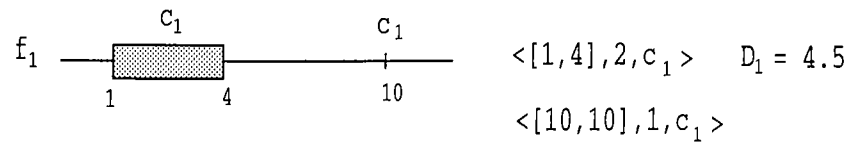


Figure 3.2. An example of construction of the intervals in the COFI Algorithm with the order of the training instances is changed.

given as shown in Figure 3.1. Then the construction of the intervals will be as shown in Figure 3.1. Here generalization distance D_1 is updated according to generalization ratio g and the current maximum and minimum feature values. It is clear that current maximum and minimum feature values depend on the order of the training instances. It is seen that, three point intervals are constructed after the execution of the third instance in Figure 3.1.

However, if the order of the instances is changed, then the construction of the concept descriptions will be as the one shown in Figure 3.2. Here, the order of the training instances is also shown in Figure 3.2. It is seen that, two intervals are constructed after the execution of the third instance. Therefore, updating generalization distances and updating intervals are the dynamic processes of the COFI algorithm, and these processes make the COFI algorithm incremental.

3.1.4 Inductive Learning

Inductive learning is a process of acquiring knowledge by drawing inductive inference from the facts that are provided by a teacher or an environment. Such a process involves operations of generalizing, specializing, transforming, correcting and refining knowledge representations. There are several different methods by which a human or machine can acquire knowledge such as rote learning (or learning by being programmed), learning from instruction (or learning by being told), learning from teacher provided examples (concept acquisition), and learning by observing the environment and making discoveries (learning from observation and discovery).

The COFI algorithm performs the learning task from a set of preclassified training examples and makes generalizations on the knowledge to construct the concept descriptions.

3.1.5 Domain Dependence in Learning

Domain dependence is important for some learning strategies. For example, explanation-based generalization (EBG) requires considerable amount of domain specific knowledge to construct explanations. In explanation-based learning (EBL), domain specific knowledge is applied to formulate valid generalizations from a single training example. The characteristics common to these methods is their ability to explain why the training instance is a member of the concept being learned [11, 39].

Although there are common principles underlying explanation-based generalization, implementations are, of necessity, domain dependent. This results from the fact that explanation-based systems must construct *explanations* each time they experience a prediction failure, and these explanations always draw upon domain knowledge. As a result, every explanation-based learning system is different since every domain has its own special knowledge.

Exemplar-based learning, on the other hand, does not construct explanations at all. Instead, it incorporates new examples into its experience by modifying its existing concept representation in the memory. Because it does not convert examples into another representation form, it does not need a domain theory to explain what conversions are legal. A consequence of domain independence is that systems can be adapted to new domains quickly without any extra domain knowledge.

The COFI algorithm is a kind of exemplar-based learning system. There is only one domain specific parameter in the COFI algorithm: *generalization ratio*. According to the structure of the features, this parameter is externally set.

3.1.6 Number of Concepts and Number of Features

Many early programs could learn exactly one concept, and some systems still remain limited in that sense. The COFI algorithm handles multiple concepts. Since it allows overlapping among classes (concept descriptions), the number of classes is not very important for the execution of the COFI.

Number of features also is not important for the COFI algorithm. It can handle large number of features. Usually learning programs have difficulty to handle large number of features. The problem is that with a large number of features, very little can be learned using concept learning techniques. Exemplar-based learning systems perform best with a small number of variables, as do many other learning techniques.

3.1.7 Properties of Feature Values

The attributes in a dataset may be nominal (categorical), or they may be continuous (numerical). The term continuous is used in literature to refer to attributes taking on numerical values (integer or real), or in general an attribute with a linearly ordered range of attribute values.

Usually learning programs handle only binary variables or only continuous variables, but not both. The learning techniques that use only nominal attribute values divide range of continuous values into subrange of values and assign each subrange to a nominal value, discarding the linear order defined on these values. The COFI algorithm handles features which have any type of values. Linear features may take on values from $-\infty$ to ∞ and they are continuous. Nominal features take on discrete feature values, for example, color attribute of an object is a nominal feature, or binary values such as answers to yes/no questions are also nominal feature values.

The COFI algorithm handles both the linear and nominal values in the same way but the generalization distances are taken as zero for the nominal values, because nominal feature values are not generalized.

3.2 Description of the COFI Algorithm

In this section the detailed description of the voting-based learning from overlapping feature intervals will be presented and illustrated.

Two processes of the COFI algorithm will be explained in the following sections: Training and the prediction processes. In the training of the COFI

algorithm, learning task is performed by learning the knowledge representation *overlapping feature intervals*. In the prediction process, a test instance is taken and a prediction is made for its class value, that is, the prediction phase corresponds to classification by predicting the class of an unseen instance. In the following two subsections, the training and the prediction algorithms will be explained in details through examples.

3.2.1 Training in the COFI Algorithm

The input of the COFI algorithm is a training dataset. Each instance in the training dataset is represented by a vector. This vector's size is equal to the number of features plus one for the associated class value. That is, i th instance e is represented as $e_i = \langle x_{i,1}, x_{i,2}, \dots, x_{i,n}, c_j \rangle$ where $x_{i,1}, x_{i,2}, \dots, x_{i,n}$ are the corresponding feature values of features f_1, f_2, \dots, f_n , and c_j is the associated class of the example e_i where $1 \leq j \leq k$, here k is the total number of the classes. Therefore, the dimension of the example vector e_i is $n + 1$ where n is the number of features. In the COFI algorithm, both nominal and continuous feature values can be processed. In the algorithm, the continuous valued features are referred as linear features, and nominal valued features are referred as nominal features.

In the training process, examples are processed one by one and the corresponding intervals are constructed. This construction is described in detail and illustrated below. After the training phase, the COFI algorithm has completed its learning task. In other words, the COFI algorithm has learned the projections of the class intervals of each class dimension for each feature.

Learning overlapping feature intervals is done by storing the objects separately in each class dimension for each feature as class intervals of values. Basic unit of the representation is the “*interval*” in this algorithm. An interval consists of four parameters; lower and upper bounds, representativeness count and a class value. Lower and upper bounds of the interval are the minimum and maximum feature values that fall into the interval respectively. Representativeness count is the number of the instances that the interval represents, and finally the class value is the associated class of the interval. The COFI algorithm performs the learning task by learning the projection of the concepts

over each class dimension for each feature, that is, the COFI algorithm learns the overlapping feature intervals for each feature.

Here, the training algorithm of the COFI will be presented and illustrated. Instances are represented as a vector of feature values and the associated class value. Initially, when the first example is processed, a point interval is constructed. This interval's lower and upper bounds are equal and this boundary value is the corresponding feature value. It is represented as a point in the corresponding class's feature value.

As an example, suppose that the first training instance of a training dataset e_1 is of class c_1 . As shown in Figure 3.3a, the corresponding class dimensions are updated and a point interval is constructed for each class dimension of each feature. In other words, if the first example is $e_1 = \langle x_{1,1}, x_{1,2}, c_1 \rangle$ where $x_{1,1}, x_{1,2}$ are the feature values of f_1, f_2 and c_1 is the associated class, then a point interval will be constructed for the corresponding class dimension of each feature. In fact, a point interval firstly partitions the related class dimension of each feature into three intervals: First interval's lower bound is $-\infty$, upper bound is $x_{1,1}$ and representativeness count is 0, second interval's lower and upper bound are $x_{1,1}$ and the representativeness count is 1, finally the third and the last interval's lower bound is $x_{1,1}$, upper bound is ∞ and the representativeness count is 0 again. First and third interval's class value is associated as undetermined but the second interval's class value is determined by the related instance's class value, which is c_1 . Here an interval is represented as $\langle [lower\ bound, upper\ bound], class\ value, representativeness\ count \rangle$.

According to this representation, the intervals for feature f_1 in Figure 3.3a may be represented as follows:

the first interval is :

$$\langle [-\infty, x_{1,1}), \text{UNDETERMINED}, 0 \rangle,$$

the second interval is:

$$\langle [x_{1,1}, x_{1,1}], c_1, 1 \rangle,$$

the third interval is :

$$\langle (x_{1,1}, \infty], \text{UNDETERMINED}, 0 \rangle .$$

For the second feature f_2 , similar representation is valid. In this case, the

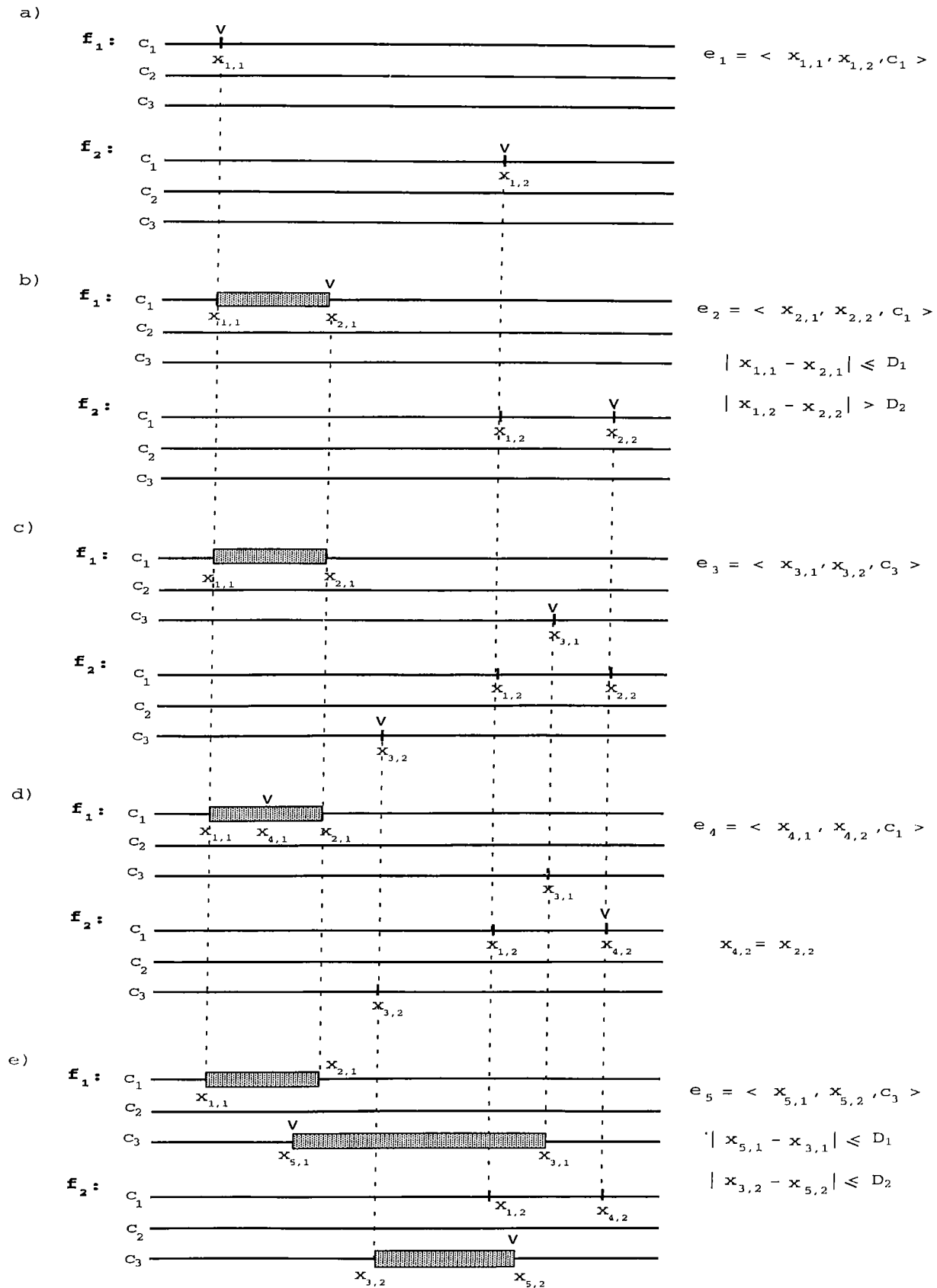


Figure 3.3. An example of construction of the intervals in the COFI Algorithm.

```
update_generalization_distance(feature f):  
begin  
  D_f = (current_max(f) - current_min(f)) * generalization_ratio  
end.
```

Figure 3.4. Updating generalization distances in the COFI Algorithm.

feature value is $x_{1,2}$. If the $x_{1,1}$ is replaced by $x_{1,2}$ in the above representation, then the second feature's interval representations will be provided.

If the next coming training example belongs to another class dimension, the same procedure is applied and a new point interval is formed in this new corresponding class dimension. However, if it belongs to the same class dimension, then there is a potential for generalization of intervals. An interval may be extended through generalization with other neighboring points or the intervals in the same class dimension to cover the new feature value. In order to generalize the new feature value, the distance between this value and the previously constructed intervals must be less than the generalization distance, D_f , for this feature.

Estimating the generalization distance D_f for each feature is the first task of the training process. The algorithm of this process is given in Figure 3.4. Here the current maximum and current minimum feature values are the maximum and minimum values of the related feature seen up to the current example. They are updated by each new training example. The `current_max(f)` function in Figure 3.4, returns the maximum value for feature f and the function `current_min(f)` returns the minimum value for feature f encountered up to the current training instance. The algorithm works in an incremental manner. Since `current_max(f)` and `current_min(f)` change through out the training process, the COFI algorithm is effected by the order of the training instances. Firstly, the maximum and the minimum values are equal to each other and they are the first feature value of the related feature of the training instance. Therefore, initially all the generalization distances are 0 for each feature.

If the feature values of the next training instance are different from the previous example's feature values, then one of the maximum and minimum

value of the related feature is updated so the generalization distance will also be updated. The change in the generalization distances D_f are computed before updating the class intervals. *Generalization ratio*, g , is an external parameter and used to scale the generalization distances. The generalization distance is computed as given below if the generalization ratio is g (the generalization ratio g is in the range $[0,1]$).

$$D_f = (\text{current_max}_f - \text{current_min}_f) * g. \quad (3.1)$$

The generalization process is applied only to linear type of features. Nominal feature values are not generalized. Therefore, if nominal feature values are processed then no generalization is done and D_f values are taken as 0 for these kinds of features. The algorithm of the updating the generalization distances for each feature is given in Figure 3.4. The training process of the COFI algorithm is given in Figure 3.5.

After deciding the generalization distance D_f , the intervals should be updated according to D_f . If the distance between the new coming example's feature value and the previously constructed intervals is greater than the D_f , then the new coming example constructs a new point interval.

Suppose that the second training instance e_2 's first feature value is $x_{2,1}$ and the distance between $x_{2,1}$ and the point interval $\langle [x_{1,1}, x_{1,1}], c_1, 1 \rangle$ is less than the generalization distance D_1 . The second instance e_2 's second feature value is $x_{2,2}$ and the distance between $x_{2,2}$ and the point interval $\langle [x_{1,2}, x_{1,2}], c_1, 1 \rangle$ is greater than the generalization distance D_2 . That is, for the second example $e_2 = \langle x_{2,1}, x_{2,2}, c_1 \rangle$,

$$|x_{1,1} - x_{2,1}| \leq D_1 \quad \text{and}$$

$$|x_{1,2} - x_{2,2}| > D_2.$$

For the first feature f_1 , generalization will occur, in the f_1 's corresponding class dimension c_1 , but no generalization will be done on f_2 's class dimension c_1 . Example of this process is shown in Figure 3.3b. The COFI algorithm will generalize the interval for $x_{1,1}$ into an extended interval $\langle [x_{1,1}, x_{2,1}], c_1, 2 \rangle$. Here, the representativeness count is also incremented by one, because this interval represents two examples now. Another point interval is constructed in

```
train(training dataset) :
begin
  foreach example tr_e in the training dataset
    foreach feature f
      update_generalization_distance(f)
      foreach class c
        update_intervals
end.
```



```
update_intervals(feature value of f):
begin
  key = feature value of f
  search interval that the key fall into
  if (interval = NULL) then
    make new point interval
  elseif (interval->lower > key) then
    if (interval->lower - key) <= generalization_distance then
      interval->lower = key
      interval->representativeness++
    else
      make new point interval
  elseif (interval->upper >= key) then
    interval->representativeness++
  elseif (key - interval->upper) >= generalization_distance then
    if next interval is close enough
      join two nodes
    else
      make new point interval
end.
```

Figure 3.5. Training process in the COFI Algorithm.

the f_2 's corresponding class dimension, so this feature's c_1 dimension will have two intervals $\langle [x_{1,2}, x_{1,2}], c_1, 1 \rangle$ and $\langle [x_{2,2}, x_{2,2}], c_1, 1 \rangle$.

If the new training example falls into an interval with a different class then the same procedures are executed for this new class dimension. If for example, third training instance e_3 's class is c_3 , then the partitioning will be done in the c_3 's class dimension as in Figure 3.3c. This property of the algorithm allows overlapping, because at the same time, different class intervals may be formed for the same feature values.

If one of the feature values of the next training example falls into an interval properly as shown in Figure 3.3d, then the interval's representativeness count is incremented by one, that is if the interval is $\langle [x_{1,1}, x_{2,1}], c_1, 2 \rangle$ and the next instance's first feature value, $x_{3,1}$, is between $x_{1,1}$ and $x_{2,1}$, that is, $x_{1,1} \leq x_{3,1} \leq x_{2,1}$, then the new interval will be $\langle [x_{1,1}, x_{2,1}], c_1, 3 \rangle$. No physical change occurs on the boundaries of the interval but the representativeness count is incremented by one.

Overlapping Concept Descriptions

Perhaps the most important feature of the COFI algorithm is that it allows overlapping of intervals which are generalizations of feature values. That is, it has the ability to form different class or concept definitions for the same feature values. The rationale of this approach is based on the fact that there may exist different class values for the overlapping feature values. The COFI algorithm may store many intervals which correspond to same feature values but different class values. Here it is assumed that there are disjunctive concepts. Many concept learning programs have ignored disjunctive concepts, because they are quite difficult to learn.

Let us show the overlapping of concept descriptions intervals through an example. Let the fifth training instance of Figure 3.3 belongs to c_3 . The feature values of this instance is shown in Figure 3.3e. Assume that the generalization distances D_1 is greater than the distance between $x_{3,1}$ and $x_{5,1}$, so a range interval is constructed for the first feature's c_3 class dimension. Similarly, D_2 is greater than the distance between $x_{3,2}$ and $x_{5,2}$, again a range interval is constructed for the second feature's c_3 class dimension. Here, the overlapping

occurs between the descriptions of class c_1 and c_3 . For the feature f_1 , overlapping occurs between the feature values $x_{5,1}$ and $x_{2,1}$ and for the feature f_2 overlapping occurs at the point $x_{1,2}$. In Figure 3.3e, it is seen that, in the COFI algorithm no specialization is done, that is, there is no subdivision of existing intervals. This approach is plausible, because in real life, different concepts (classes) may exist at the same time, especially in medical field.

3.2.2 Prediction in the COFI Algorithm

After training with the training dataset, the prediction phase can be executed. The classification principle of the COFI algorithm is based on a majority voting taken among the individual predictions based on the votes of the features. The vote of a feature is based solely on the value of the test instance for that feature. The vote of a feature is not for a single class but rather a vector of votes, called *vote vector*. The size of the vector is equal to the number of classes. An element of the vote vector represents the vote that is given by the feature to the corresponding class. The vote that a feature gives to a class is the relative representativeness count of the class interval. The relative representativeness count is the ratio of the representativeness count to the number of examples of the corresponding class value. Since most of the datasets are not distributed normally in terms of their class values, this kind of normalization is required. The vote vectors of each feature are added to determine the predicted class. The class which receives the maximum vote is the final class prediction for the test instance.

For a given test instance i , the feature value i_f is searched on the intervals for feature f . If i_f falls properly within a determined ¹ interval in one or more of the class dimension(s), then votes of feature f_1 represented by the vote vector \mathbf{v}_f of votes $v_{f,j}$ is obtained for each feature:

$$\mathbf{v}_f = \langle v_{f,1}, v_{f,2}, \dots, v_{f,k} \rangle . \quad (3.2)$$

¹non-UNDETERMINED: has a class value

Here, k is the number of classes and

$$v_{f,c} = \begin{cases} \text{relative representativeness count} & \text{if } i_f \text{ falls in a class } c \\ \text{of the interval} & \text{on feature } f \\ 0 & \text{otherwise.} \end{cases} \quad (3.3)$$

The prediction process is performed for each feature by obtaining $v_{f,j}$ vectors and then the final vote vector of the classes is obtained by summing these vectors:

$$\mathbf{v} = \sum_{f=1}^n \mathbf{v}_f = \langle v_1, v_2, \dots, v_c, \dots, v_k \rangle. \quad (3.4)$$

where

$$v_i = \sum_{c=1}^k v_{f,c}. \quad (3.5)$$

The class which received the highest amount of votes is determined to be the predicted class for the test instance i . That is,

$$\text{prediction}(i) = c \text{ such that } v_c > v_j \text{ for each } j \neq c. \quad (3.6)$$

If the feature value i_f does not fall into any interval in any class dimension, then the class is predicted as **UNDETERMINED** and no votes (in other words 0 votes) are assigned to the vote vector elements. That is, no prediction is made for this feature value. The algorithm of the prediction process is given in Figure 3.6.

As an example of the prediction process, suppose that the intervals are formed for all the training set elements as shown in Figure 3.7. For feature f_1 , there exists one interval for first two class dimensions, and two intervals for class c_3 . The corresponding intervals are also shown in Figure 3.7. For the second feature f_2 , there exist two intervals in the first class dimension, and one of them is a point interval.

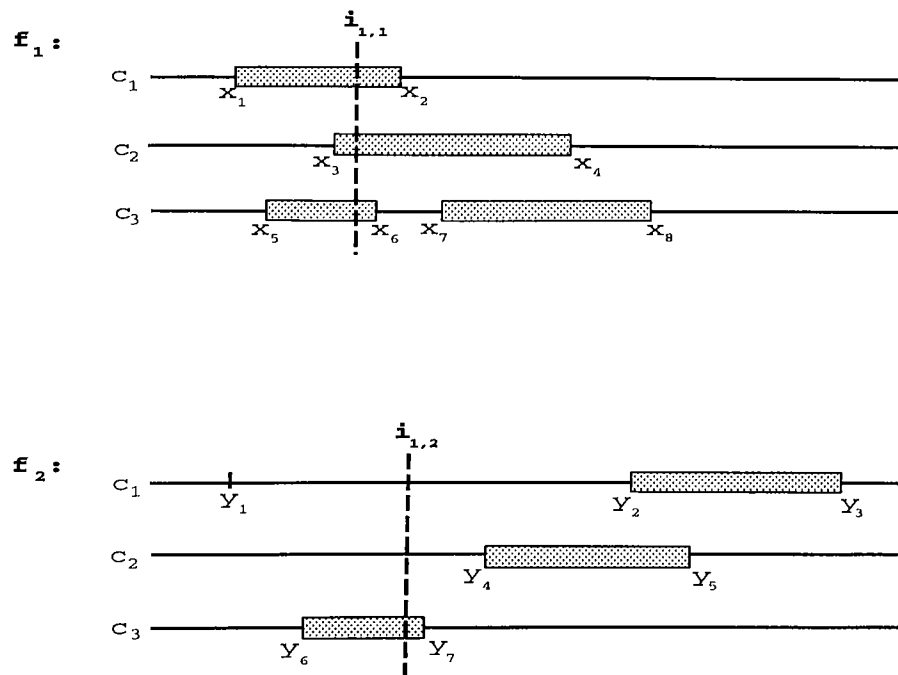
In the prediction phase, there are two possibilities. A test instance i may fall into an interval or not. For the first case, the corresponding value of the vote vector elements are the relative representativeness counts of the matched interval. Let the test instance i 's first and second feature values $i_{1,1}$ and $i_{1,2}$ be

```
prediction(test example i):
begin
  vote_vector[] = 0
  foreach feature f
    foreach class c
      interval= search(i_f)
      if (interval != NULL)
        prediction = representativeness count
                      / class count of test example i
        vote_vector[c] = vote_vector[c] + prediction

  winner = 0
  foreach class c
    if vote_vector[c] > vote_vector[winner] then
      winner = c
  if (vote_vector[winner] = 0 then
    winner = UNDETERMINED

  return(winner)
end.
```

Figure 3.6. Prediction process in the COFI Algorithm.



Formal representation of the intervals

$$\begin{aligned}
 F_1: & \langle [x_1, x_2], c_1, 100 \rangle \\
 & \langle [x_3, x_4], c_2, 80 \rangle \\
 & \langle [x_5, x_6], c_3, 75 \rangle, \langle [x_7, x_8], c_3, 20 \rangle
 \end{aligned}$$

$$\begin{aligned}
 F_2: & \langle [y_1, y_1], c_1, 30 \rangle, \langle [y_2, y_3], c_1, 70 \rangle \\
 & \langle [y_4, y_5], c_2, 80 \rangle \\
 & \langle [y_6, y_7], c_3, 95 \rangle
 \end{aligned}$$

Figure 3.7. An example of execution of the prediction process in the COFI Algorithm.

given as shown in Figure 3.7. To form the vote vectors, firstly relative representativeness count of each matched interval should be computed. As seen in Figure 3.7,

class count of c_1 : 100

class count of c_2 : 80

class count of c_3 : 95

For this case, the vote vector of the first feature is

$$\mathbf{v}_1 = \langle 100/100 \ 80/80 \ 75/95 \rangle = \langle 1 \ 1 \ 0.79 \rangle,$$

the vote vector \mathbf{v}_2 of the second feature is

$$\mathbf{v}_2 = \langle 0/100 \ 0/80 \ 95/95 \rangle = \langle 0 \ 0 \ 1 \rangle .$$

The vote vectors of each features are added to reach a final prediction, then the resulting vector \mathbf{v} is

$$\mathbf{v} = \mathbf{v}_1 + \mathbf{v}_2 = \langle 1 \ 1 \ 0.79 \rangle + \langle 0 \ 0 \ 1 \rangle = \langle 1 \ 1 \ 1.79 \rangle .$$

Here the voting is done for the maximum value, so the final prediction is the class c_3 .

3.2.3 Handling Missing Attribute Values

One important issue about the COFI algorithm is the handling of the unknown (missing) feature values. If the dataset contains unknown feature values, nothing is done about these feature values, that is, no classification or prediction is made by the COFI algorithm. This is a natural approach because in real life if nothing is known about a feature, it is usually ignored. Also in prediction phase, if an unknown value is encountered then no prediction is made, that is, the vote given to the vote vector is 0 for this feature. This is one advantage of the COFI algorithm because it is a simple and natural way.

If all class dimensions give no prediction, then no prediction is made and the resulting decision for the class is UNDETERMINED. An example of this situation

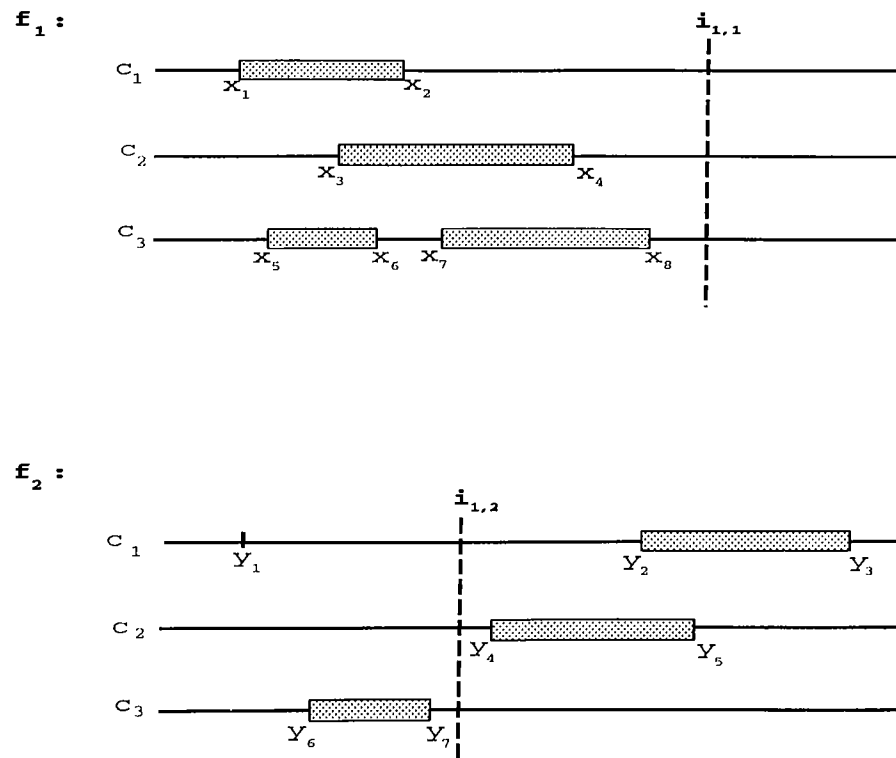


Figure 3.8. An example of execution of the prediction process in the COFI Algorithm when test instance does not fall into any intervals.

is given in Figure 3.8. Here, the test instance's first and second feature values $i_{1,1}$ and $i_{1,2}$ does not fall into any intervals, so all elements of the vote vector are 0. This results in no prediction, and the COFI algorithm gives UNDETERMINED as its prediction.

3.2.4 From Intervals to Rules

In the artificial intelligence literature, knowledge representation in the form of *if-then rules* is a well-known knowledge representation technique. Intervals constructed by the COFI algorithm can easily be converted to the if-then rules form.

The condition of an if-then rule is a conjunction of conditions about each feature value. There will be one condition for each feature and depending on the relative representativeness count, different classes can be selected. Therefore, the maximum number of if-then rules is:

$$k \prod_f ndi(f). \quad (3.7)$$

Here k is the number of classes, and $ndi(f)$ represents the total *number of disjoint intervals* formed on feature f .

Let us illustrate the conversion of intervals into if-then rules through an example. Let the representation of concepts in the form of intervals be as in Figure 3.9. Here two classes c_1 and c_2 are defined in terms of two features f_1 and f_2 . The representation in Figure 3.9 consists of only twelve intervals. However, there are five disjoint intervals in each feature. Since there are two classes, the maximum number of if-then rules is $2 * 5 * 5 = 50$. Fortunately, many of the rules can be grouped into more compact rules by connecting the conditions of the rules with the same class value.

The concept description of Figure 3.9 can be converted into eighteen if-then rules and these rules are presented in the following pages.

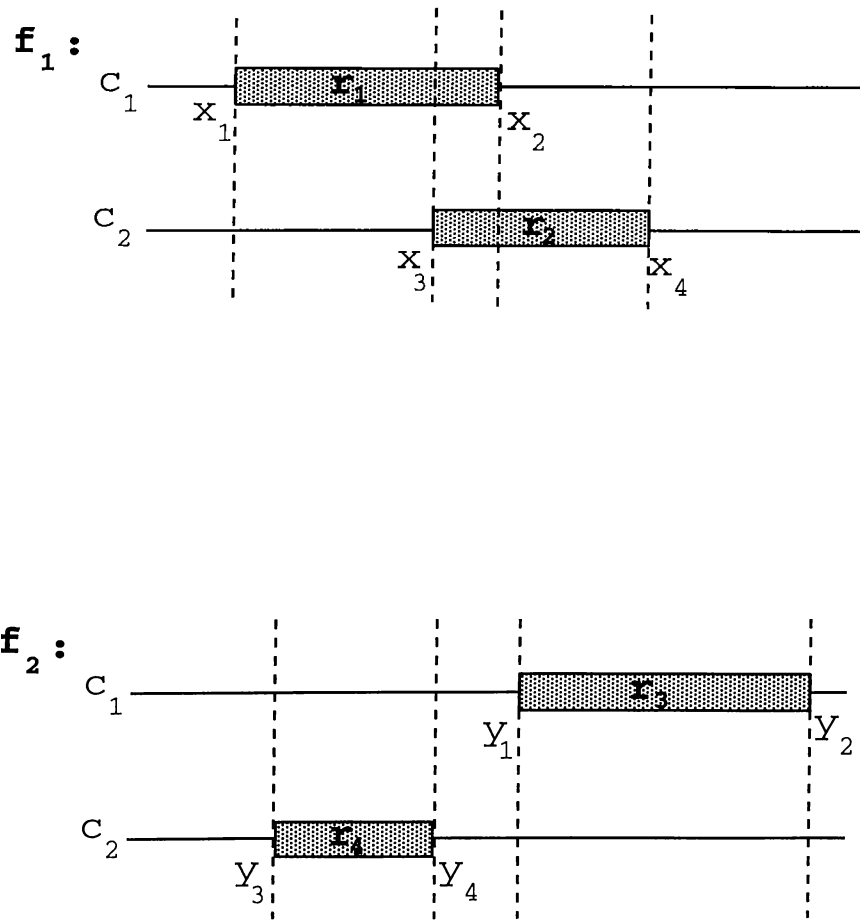


Figure 3.9. An example concept description of the COFI Algorithm.

if $f_1 < x_1$ AND ($f_2 < y_3$ OR $y_4 < f_2 < y_1$ OR $f_2 > y_2$) then

UNDETERMINED

if $f_1 < x_1$ AND ($y_3 \leq f_2 \leq y_4$) then

c₂

if $f_1 < x_1$ AND ($y_1 \leq f_2 \leq y_2$) then

c₁

if $x_1 \leq f_1 \leq x_3$ AND ($f_2 < y_3$ OR $y_4 < f_2$) then

c₁

if $x_1 \leq f_1 \leq x_3$ AND ($y_3 \leq f_2 \leq y_4$ OR $r_1 \geq r_4$) then

c₁

if $x_1 \leq f_1 \leq x_3$ AND ($y_3 \leq f_2 \leq y_4$) then

c₂

if $x_3 \leq f_1 \leq x_2$ AND ($f_2 < y_3$ OR $y_4 < f_2 < y_1$ OR $f_2 > y_2$) AND $r_1 \geq r_2$ then

c₁

if $x_3 \leq f_1 \leq x_2$ AND ($f_2 < y_3$ OR $y_4 < f_2 < y_1$ OR $f_2 > y_2$) then

c₂

if $x_3 \leq f_1 \leq x_2$ AND ($y_3 \leq f_2 \leq y_4$ OR $r_1 \geq r_2 + r_4$) then

c₁

if $x_3 \leq f_1 \leq x_2$ AND ($y_3 \leq f_2 \leq y_4$) then

c₂

if $x_3 \leq f_1 \leq x_2$ AND ($y_1 \leq f_2 \leq y_2$ OR $r_2 \leq r_1 + r_3$) then

c₁

if $x_3 \leq f_1 \leq x_2$ AND ($y_1 \leq f_2 \leq y_2$) then

c₂

if $x_2 \leq f_1 \leq x_4$ AND ($f_2 < y_1$ OR $f_2 > y_2$) then

c₂

if $x_2 \leq f_1 \leq x_4$ AND ($y_1 \leq f_2 \leq y_2$ OR $r_3 \geq r_2$) then

c₁

if $x_2 \leq f_1 \leq x_4$ AND ($y_1 \leq f_2 \leq y_2$) then

c₂

if $f_1 > x_4$ AND ($f_2 < y_3$ OR $y_4 < f_2 < y_1$ OR $f_2 > y_2$) then

UNDETERMINED

if $f_1 > x_4$ AND $(y_3 \leq f_2 \leq y_4)$ then

c_2

if $f_1 > x_4$ AND $(y_1 \leq f_2 \leq y_2)$ then

c_1

In this system, the rules are executed in the specific order given above. They are executed one by one and if the current rule's condition is true then the prediction is made as the antecedent of that rule and the execution is stopped, that is, other following rules are not tried. For example, the rule

if $x_1 \leq f_1 \leq x_3$ AND $(y_3 \leq f_2 \leq y_4$ OR $r_1 \geq r_4)$ then

c_1

is executed before the rule

if $x_1 \leq f_1 \leq x_3$ AND $(y_3 \leq f_2 \leq y_4)$ then

c_2 .

For this example, if the condition of the former rule is true then other rules will not be executed and the class value will be predicted as c_1 . However, if it is false, then the latter rule will be executed and if its condition is true then c_2 will be predicted as the class value. Here, it should be noted that, since the execution is done in the given specific order, the inverse of the condition $r_1 \geq r_4$, namely, $r_1 < r_4$ is not added as a condition to the latter rule.

3.3 Comparison of COFI with Other Methods

An instructive comparison of the COFI algorithm may be made with the CFP algorithm. CFP is the basis of this voting based algorithm. The representation and processing of the data are very similar in both techniques. Both algorithms have similar training and prediction phases. The main difference stems from the fact that the CFP algorithm does not allow overlapping intervals. In the training phase of the CFP algorithm, if a new feature value falls in an interval

with a different class than that of the example and if the segment is a point segment, then a new point segment corresponding to the new feature value is inserted next to the previously constructed segment. Otherwise, if the class of the partition is not undetermined, then the CFP algorithm specializes the existing partition by dividing it into two sub-partitions and inserting a point segment corresponding to the new feature value in between them. Here it is seen that the CFP works on feature dimensions for each class and make specializations. However, in the COFI, no specialization is made by the algorithm. Since the class dimensions are different for each feature, there is no problem about a feature value which falls in an interval with a different class.

In the CFP algorithm, weight assignment and adjustment is done for each feature. The training process in CFP has two steps: learning feature weights and learning feature segments. In COFI, the training process is again made of two phases, but the first phase is updating the generalization distances. Weight assignment is done through representativeness count to the intervals, not to the whole feature.

CFP is much more sensitive to irrelevant features than the COFI algorithm. Since the CFP algorithm allows specializations on the feature dimension, many number of unwanted segments are constructed when irrelevant features are processed. However, COFI makes no specializations. Therefore, when irrelevant attributes are processed, usually one interval is constructed for each class dimension and it covers whole class dimension. By using this property of the COFI algorithm, irrelevant attributes can easily be detected. In the prediction process, vote vectors of all class dimensions give their relative representativeness count as prediction. If examples are distributed normally in terms of their class values, then their predictions will have no effect to the final prediction for the irrelevant attributes. An example of this situation will be given in Chapter 4. This is one of the advantage of the COFI algorithm over the CFP algorithm.

The CFP algorithm has much more domain dependent parameters than the COFI algorithm. The generalization distances and the weight adjustment rate are the parameters of the CFP algorithm. In the COFI algorithm, there is only one domain dependent parameter, generalization ratio, g . One advantage of using less domain dependent parameters is that the algorithm is affected less by the changes of the data. However, one disadvantage is that all features are

affected and updated by using one unique parameter. Therefore, it is not easy to set this parameter properly for each feature.

After the COFI and the CFP algorithms comparison, we will compare our new method with the NBC and the NN* algorithms, because both the NBC and the NN* methods process features independently as the COFI algorithm does.

Both the NBC and the NN* algorithms are non-incremental algorithms, however the COFI algorithm has an incremental structure. These three algorithms apply the majority voting procedure for the classification.

NBC usually gives the optimum solution for the prediction. However, it requires much more memory space than the COFI algorithm and its time complexity is also greater than the COFI algorithm.

The performance of the NN* is usually worse than others, besides, its memory requirement and time complexity is greater than the COFI algorithm. In Chapter 4, we will present the comparison results of the NBC, the NN* and the COFI algorithms in terms of accuracy and memory requirement on artificial and real-world datasets.

It is instructive to compare our COFI algorithm with the 1R rule, as well. 1R, defined by Holte in 1991 [26], can be treated as a special case of the COFI algorithm. In this technique, as in the COFI algorithm, feature intervals are constructed to form concept descriptions. All classes and features are processed independently, as processed in the COFI algorithm. In 1R, after the training, one of the feature's concept description is taken as the *rule*. All prediction process depends on this selected feature's concept description. However, in the COFI algorithm, after the training phase, a majority voting is done among the concept descriptions of all features.

As a final word for this section, we should emphasize the facts that the COFI, the CFP, the NBC and the NN* algorithms process each feature independently in prediction and all of these algorithms select the class that receives the highest amount of votes from features.

Chapter 4

Evaluation of the COFI Algorithm

In this chapter, both empirical and theoretical evaluations of the COFI algorithm will be presented. In the theoretical evaluation, the training and the testing time complexities of the COFI algorithm are computed. In the empirical evaluation, COFI algorithm is compared with the NN*, the NBC and the CFP algorithms on artificial and seven real-world datasets.

4.1 Theoretical Evaluation of the COFI Algorithm

Intervals are constructed for each feature and class dimension in the training of the COFI algorithm. To construct the whole concept description, of course, all the training examples should be processed. This means that, we should process all feature values of each instances and put them on the corresponding class dimension.

The time complexity of the COFI algorithm depends on the number of intervals. The number of intervals is determined by the generalization ratio, the nature of attributes of the training examples and the repeated feature values of the training examples. If nominal features are processed, then no generalization is done. In this case, all the intervals are point intervals, also, if all of them have unique values then the number of intervals is equal to the number of training examples for each feature. This is a very rare situation in real world datasets, if it occurs then these kind of features are usually irrelevant

features.

Updating the current representation by a training instance requires to search the interval, that the training instance falls on for its class dimension. Determining the interval on a given dimension for a given feature value requires a search on an ordered list. The complexity of this search is proportional to the number of existing intervals currently on the corresponding feature class dimension. Let the total number of training instances be m . In the worst case the number of intervals for one feature is m . Therefore, the time required by the training process for the i th instance is

$$c_t.n.k.(i/k) \quad (4.1)$$

where c_t is the training constant, n is the number of features, k is the number of classes and i/k is the average number of intervals for one class dimension. For all the training instances, this time will be

$$c_t.m.n.k.(m/k) \quad (4.2)$$

Here, average number of intervals for one class dimension is m/k , since we have k class dimensions for each feature, there are maximum m intervals for each feature. Therefore, training time complexity of the COFI algorithm is

$$O(m^2n). \quad (4.3)$$

In the prediction, we have intervals to search for the given feature values. Therefore at the worst case, complexity of classification of a test instance is equal to the sequential search time on ordered lists which is

$$O(mn). \quad (4.4)$$

By using an appropriate data structure, for example, balanced binary tree, search can be done in $O(\log m)$. However, for its simplicity in the implementation we choose to maintain the intervals on a feature-class dimension as an ordered list.

Space complexity of the COFI algorithm is usually less than other techniques that store examples verbatim in memory. However, at the worst case, the space complexity of the COFI algorithm is

$$O(mn). \quad (4.5)$$

Here, as it is explained above, mn is the maximum total number of intervals.

4.2 Empirical Evaluation of the COFI Algorithm

In this subsection, results of the experimental evaluation of the COFI algorithm will be presented in details and compared with some other learning models. The COFI algorithm is tested on seven real-world datasets which are widely used in machine learning field, therefore comparisons are possible with other models. The real-world datasets are selected from the collection of datasets distributed by the machine learning group at the University of California at Irvine. We will compare the COFI algorithm with the NBC, the NN* and the CFP algorithms.

4.2.1 Methodology of Testing

Most definitions of learning rely on some notion of improved *performance*. Thus various performance measures are the natural dependent variables for machine learning experiments, just as they are for studies of human learning. There are three important measures for evaluation for a learning algorithm. They are *accuracy*, *time*, and *storage complexity*. In the previous section, we have computed the time and storage complexity of the COFI algorithm. In this subsection, accuracy will be measured empirically.

Several measures of performance are possible. For supervised concept learning tasks, the most commonly used metric is the percentage of correctly classified instances over all test instances. This metric cannot be used for unsupervised learning tasks like conceptual clustering, but this measure can be generalized as the average ability to predict attribute values [19].

Accuracy of an algorithm is a measure of correct classifications on a test set of unseen instances. There are several ways of measuring the accuracy of an algorithm, in the literature the common techniques are *cross-validation*, *leave-one-out* and *average of randomized runs*.

Average of Randomized Runs: This method involves testing the algorithm over randomly selected training and testing sets. These sets are disjoint. The test is repeated for a fixed number of times. The final result is the average accuracy across all trials.

Cross-Validation: This technique involves removing mutually exclusive test sets of examples from the dataset. For each test set, the remaining examples serve as a training set, and classification accuracy is measured as the average accuracy on all the test sets. The union of the all test sets equals to the whole dataset.

Leave-one-out: It is an elegant and straight forward technique for estimating classifier error rates. Because it is computationally expensive, it is often reserved for relatively small samples. For a given method and sample size (number of instances) m , a classifier is generated using $m - 1$ cases and tested on the remaining case. This is repeated m times, each time designing a classifier by leaving-one-out. Each case is used as a test case and, each time nearly all the cases are used to design a classifier.

Evidence for the superiority of the leave-one-out approach is documented in the literature [15, 31]. While leave-one-out is a preferred technique, with large samples it may be computationally expensive. As the sample size grows, traditional train and test methods improve their accuracy in estimating error [28].

In this thesis, leave-one-out technique is used to report the performance of the COFI algorithm. We have chosen this technique because the accuracy of the COFI algorithm depends on the order of the training set. Another important issue in testing is that we want to make sure the training and testing sets are disjoint. Therefore, the test is performed on an unseen example in the COFI algorithm.

4.2.2 Experiments with Artificial Datasets

The success of a learning system is very dependent on the ability to cope with noisy and incomplete data, an adequate knowledge representation scheme, having a low learning and prediction complexities and the effectiveness of the learned knowledge [34]. Most of the real-world datasets contain incomplete and inconsistent data, therefore the ability to handle the inconsistent and incomplete data is very important for learning algorithms.

Many researchers tackled the problem of handling examples which contain

error or *noise* [7, 45, 52]. There are two types of noise that can be found in real-world datasets: classification noise and attribute noise.

One type of noise is the existence of irrelevant attributes. Real-world datasets usually contain unequally relevant attributes. Since which attributes are relevant and which are irrelevant can not be known before all of them are collected, all attributes should be taken initially.

Another type of the noise is the unknown (or missing) feature or class values. Handling unknown feature values is simpler than handling the unknown class values. To solve the problem of unknown feature values several techniques were used. One of them is to ignore them as it is done in NBC, NN*, CFP and COFI. In some techniques, additional instances are generated for all possible values of the missing attribute and *rough sets theory* [55] is used to solve the conflicts. It is a costly solution and applicable to attributes which have finite number of possible values.

To test the performance of the COFI algorithm on domains which have unequally relevant attributes, we have generated artificial datasets. These datasets contain additional attributes that are randomly generated. We can control the system's behavior on the irrelevant attributes by using these artificial datasets. The artificial dataset, with no irrelevant features, contains 300 instances, four features and three classes, 100 examples for each class.

The noise-free concept descriptions used in the artificial data are given below, these are hyperrectangles in a 4-dimensional space:

class 1:

$$0.0 < f_1 < 5.0 \ \& \ 0.0 < f_2 < 2.0 \ \& \ 0.0 < f_3 < 2.0 \ \& \ 0.0 < f_4 < 5.0$$

class 2:

$$4.00 \leq f_1 < 6.0 \ \& \ 4.0 \leq f_2 < 8.0 \ \& \ 5.0 < f_3 < 7.0 \ \& \ 4.0 < f_4 < 6.0$$

class 3:

$$7.0 < f_1 < 10.0 \ \& \ 7.0 < f_2 < 10.0 \ \& \ 2.0 \leq f_3 < 4.0 \ \& \ 2.0 < f_4 < 5.0$$

Learning with Unequally Relevant Attributes:

Both in machine learning and pattern recognition fields, most of the real-world datasets contain many unequally relevant features [6, 33]. Therefore, handling of these unequally relevant features is a very important task for a learning algorithm. Examples can be found in the pattern recognition tasks in which feature detectors automatically extract a large number of features for the learner [6]. Some of these features are not as relevant as the others. Another example can be seen in the field of medicine. The medical records of a large number of patients usually contain more information than is actually required for describing each disease in the task of learning diagnostic rules for several different disease.

One of the solutions to this problem is to use feature weights as used successfully in [29, 49, 56]. A weight is assigned to each feature according to its importance. This importance is determined according to how much this feature affects the final true prediction. If a feature gives usually correct predictions then its weight will be high, however, if it usually makes wrong predictions then its weight will be low. In Chapter 2, the weight assignment of the CFP algorithm was explained briefly.

The COFI algorithm solves the problem of irrelevant features naturally. The irrelevant features constructs concept descriptions, or in other words, intervals, such that they include whole class dimension.

Due to the generalization of intervals, usually, only one interval is constructed on the class dimension of an irrelevant feature. The lower bound of this interval is the minimum feature value and the upper bound of the feature is the maximum feature value. Therefore, if the number of examples for each class is equal to each other, that is examples are normally distributed in terms of their class values in the dataset, then the vote of each class dimension is equal to each other for the irrelevant feature. Therefore, their votes can be ignored. One example of this situation is given in Figure 4.1 and in Figure 4.2. In Figure 4.1 the noise-free concept description of our artificial dataset is shown. In Figure 4.2, the concept description of our artificial dataset is shown with one irrelevant feature. It is seen that the irrelevant feature has large intervals that cover all the class dimension for each feature. In fact, it is a normal consequence of irrelevancy. This means that, feature values are

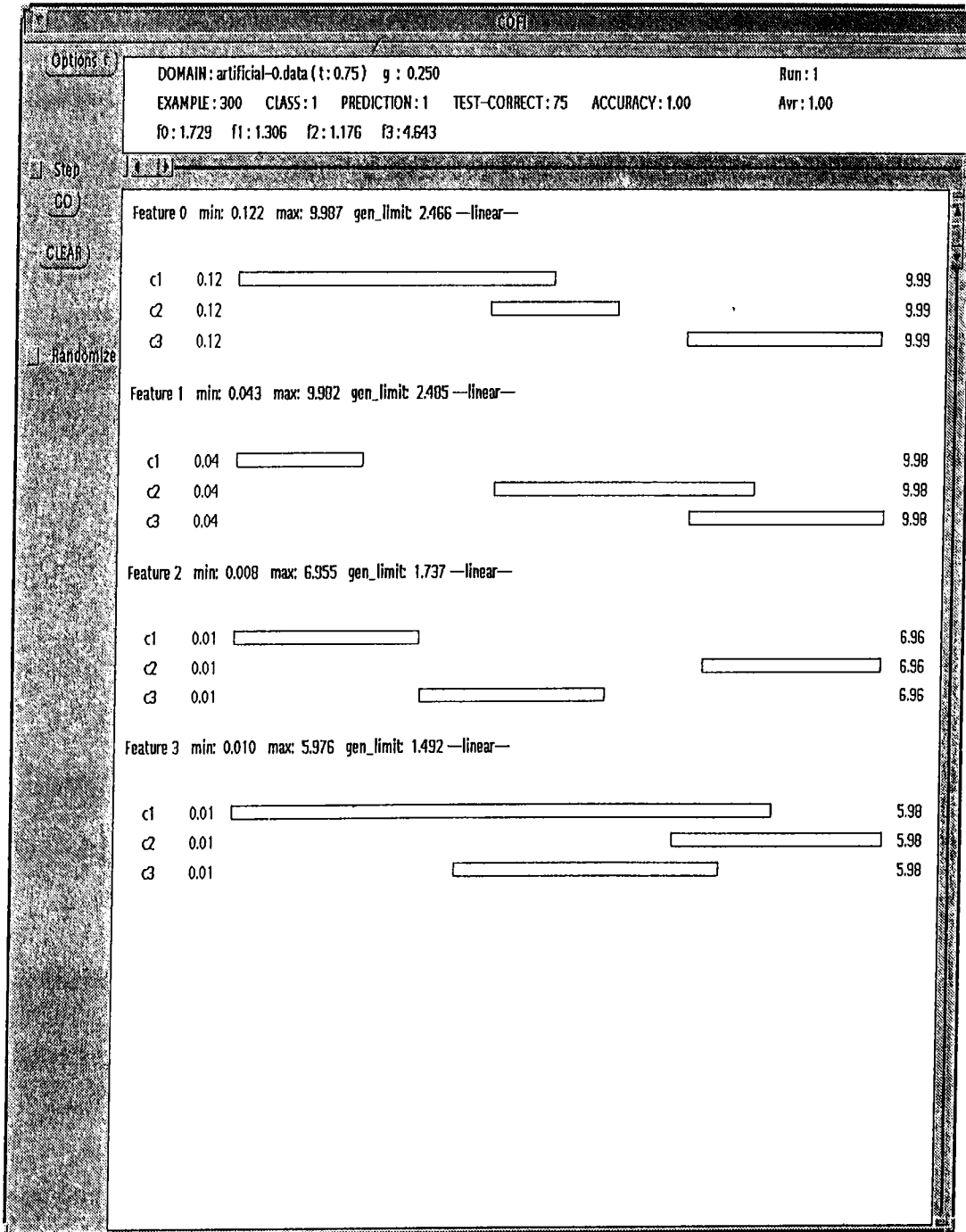


Figure 4.1. The concept description of the noise-free artificial dataset by the COFI Algorithm.

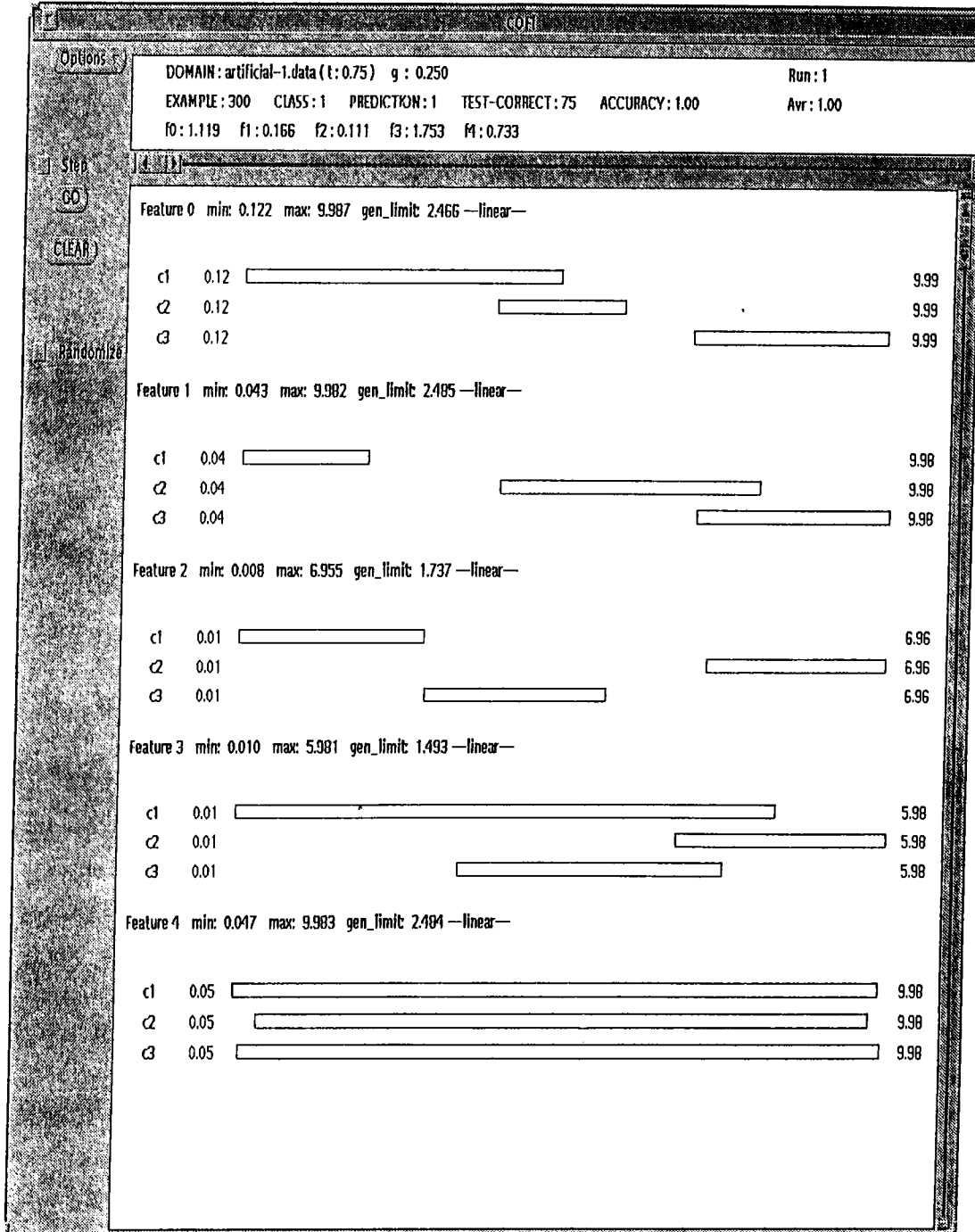


Figure 4.2. The concept description of the artificial dataset by the COFI Algorithm with one irrelevant feature.

distributed between minimum and maximum feature values without any rule or boundary through the all class dimensions. If the class counts of each class are equal, then the vote vector elements of this irrelevant feature are equal. Therefore, they can not affect the final prediction in the COFI algorithm.

Three algorithms, namely the COFI, the NN* and the NBC algorithms, are tested on eleven datasets, ten of these datasets contains irrelevant attributes. Each time an irrelevant feature is added to the previous dataset. Initial dataset is a pure (with no irrelevant feature) dataset. The effect of the irrelevant features to the accuracy results and to the memory requirement are shown in Figure 4.3 and Figure 4.4 respectively.

In Figure 4.3, it is seen that, increase in the number of the irrelevant attributes causes usually a great decrease in the accuracy for NN* algorithm. The NBC algorithm is not affected by the existence of irrelevant attributes. The COFI algorithm is also not affected by the irrelevant attributes. Its accuracy results are almost same with or without the irrelevant attribute values. This means that, the COFI algorithm is one of the most reliable algorithms when irrelevant attributes are considered.

In Figure 4.4, the success of the COFI algorithm is very clear in terms of memory requirement. In fact, an increase in the number of irrelevant attribute values causes an increase in the memory requirement for all of the three algorithms. However, since the memory requirement of the COFI algorithm is much less than the NN* and the NBC algorithms, the increase for the COFI algorithm is much less than the other algorithms. The NN* and the NBC algorithms keep all the instances in memory verbatim, so each time they need a memory space equal to the number of instances for added irrelevant attribute. However, the COFI algorithm requires a memory space equal to the number of intervals that is constructed for added irrelevant attribute. If the generalization ratio is great enough, the number of the intervals will usually be equal to the number of class for new added irrelevant attribute. Here we assumed one unit of memory is allocated for each value. Since an interval contains four elements: lower and upper bounds of the interval, representativeness count and the associated class value, the memory requirement of one interval is four units of memory.

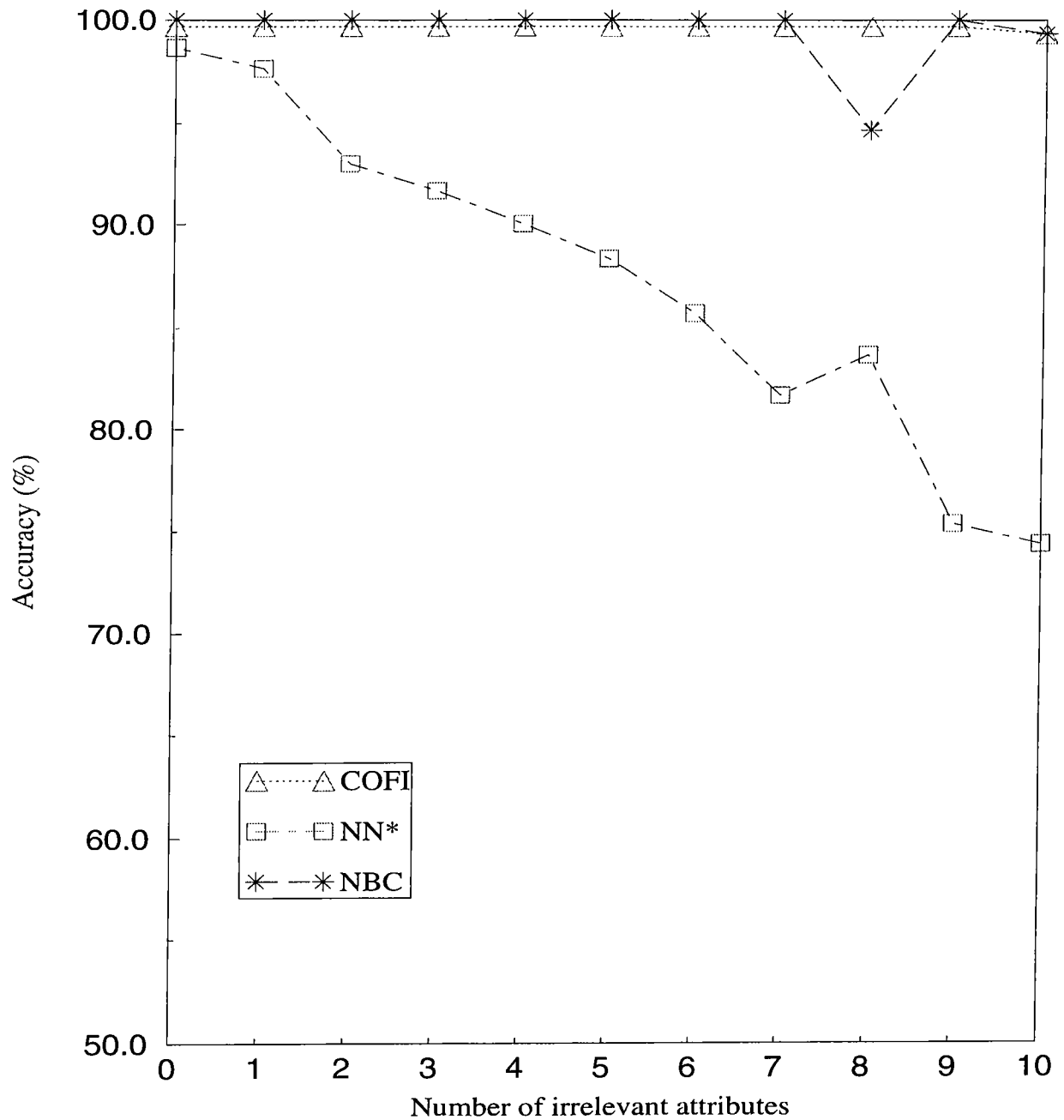


Figure 4.3. Accuracy results of the COFI, the NN* and the NBC algorithms on domains with irrelevant attributes.

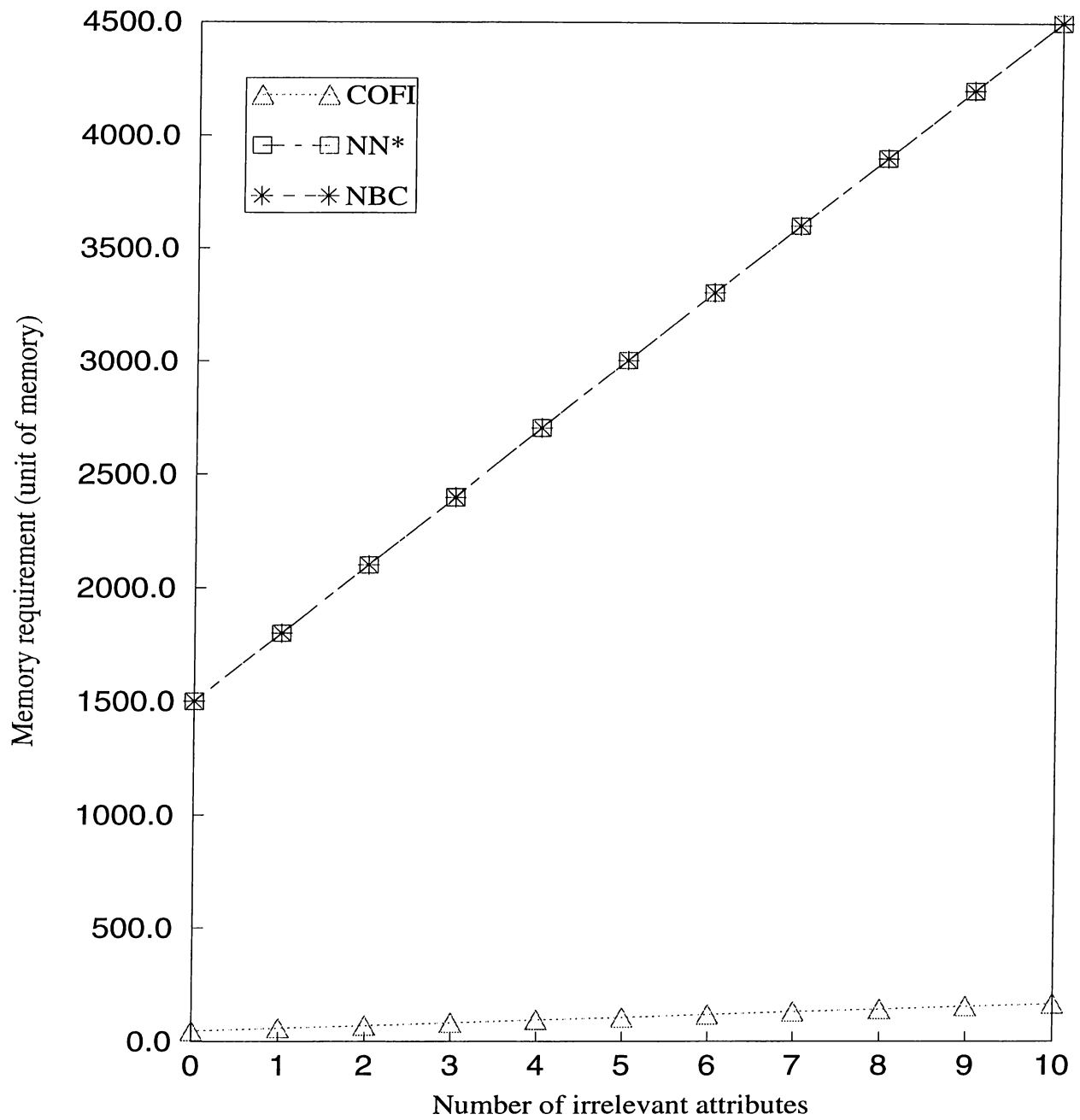


Figure 4.4. Memory requirement of the COFI, the NN* and the NBC algorithms on domains with irrelevant attributes.

Learning with Missing Attribute Values:

Every learning algorithm should handle missing attribute values in some way, because most of the real-world datasets contain unknown attribute values. Therefore, in the literature, there are some methods to handle these kinds of attribute values [23, 43, 44, 46]. Most of the methods are based on one of the following ideas:

- Ignoring examples that have unknown attribute value.
- Assuming an additional special value for unknown attribute values.
- Using probability theory by utilizing information provided by context.
- Generating additional instances for all possible values of the unknown attribute.
- Exploring all branches on decision trees, remembering that some branches are more probable than others.

The COFI algorithm selects the first method, it just ignores the unknown attribute values. This is the simplest approach, but it usually gives good results. This approach also causes reduction in training and testing time. However, other methods cause increase in the training and testing time.

Similar tests, as done for the irrelevant attributes, are done on the artificial datasets which contain missing attribute values. Figure 4.5 and Figure 4.6 show the behavior of the COFI, the NN* and the NBC algorithms in terms of accuracy and memory requirement respectively, when the unknown attribute values are increasing in the dataset.

In Figure 4.5, it is seen that the COFI algorithm is not affected by the missing attribute values at all, because this algorithm can construct nearly perfect concept description by using the known attribute values if the distribution of the known values is normal. However, the NN* algorithm is affected by the missing attribute values. Its accuracy results are steadily decreases when the number of the missing attribute values increase. The NBC algorithm is also not affected by the unknown attribute values very much such as the COFI algorithm.

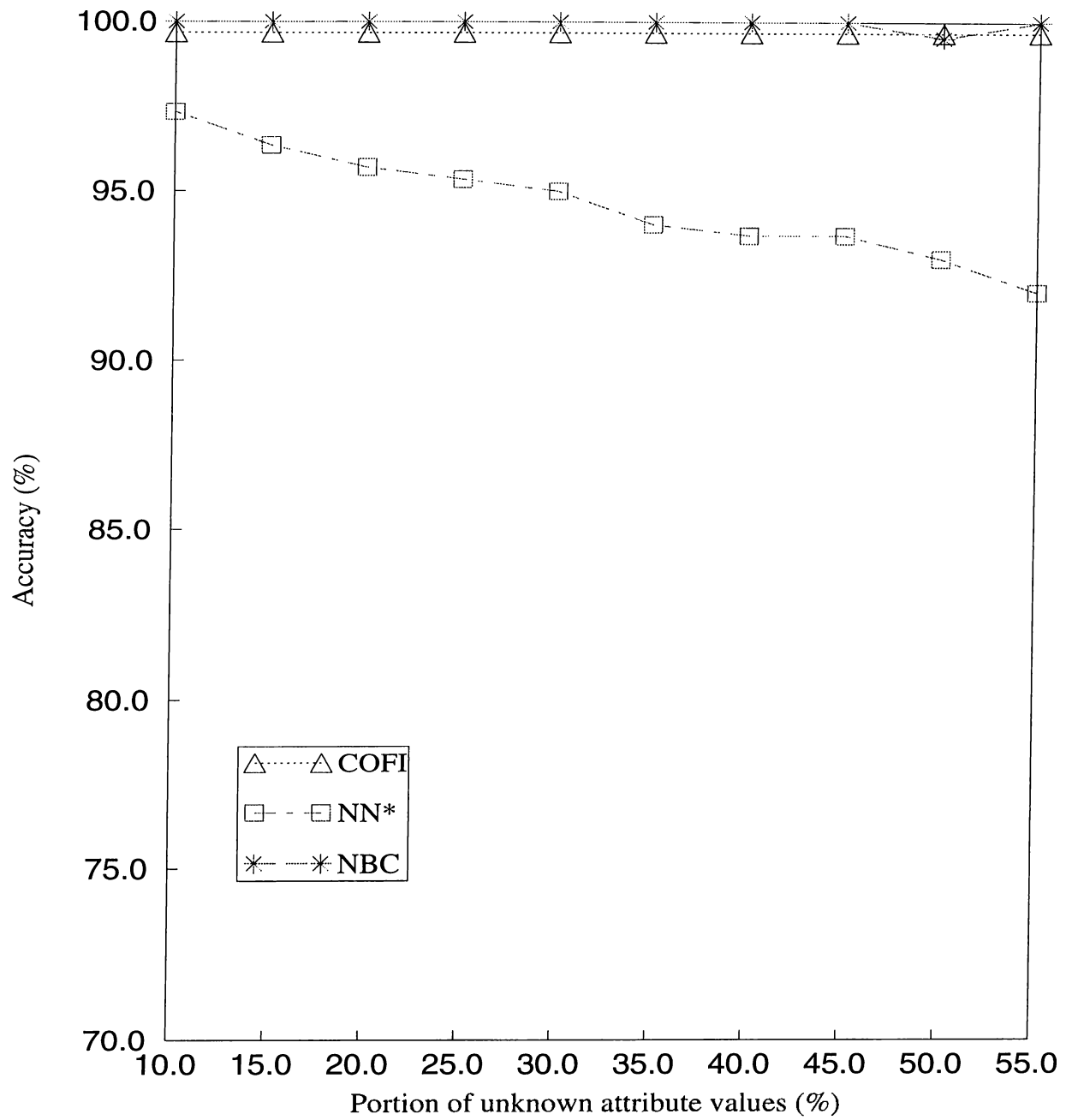


Figure 4.5. Accuracy results of the the COFI, the NN* and the NBC algorithms on domains with unknown attribute values.

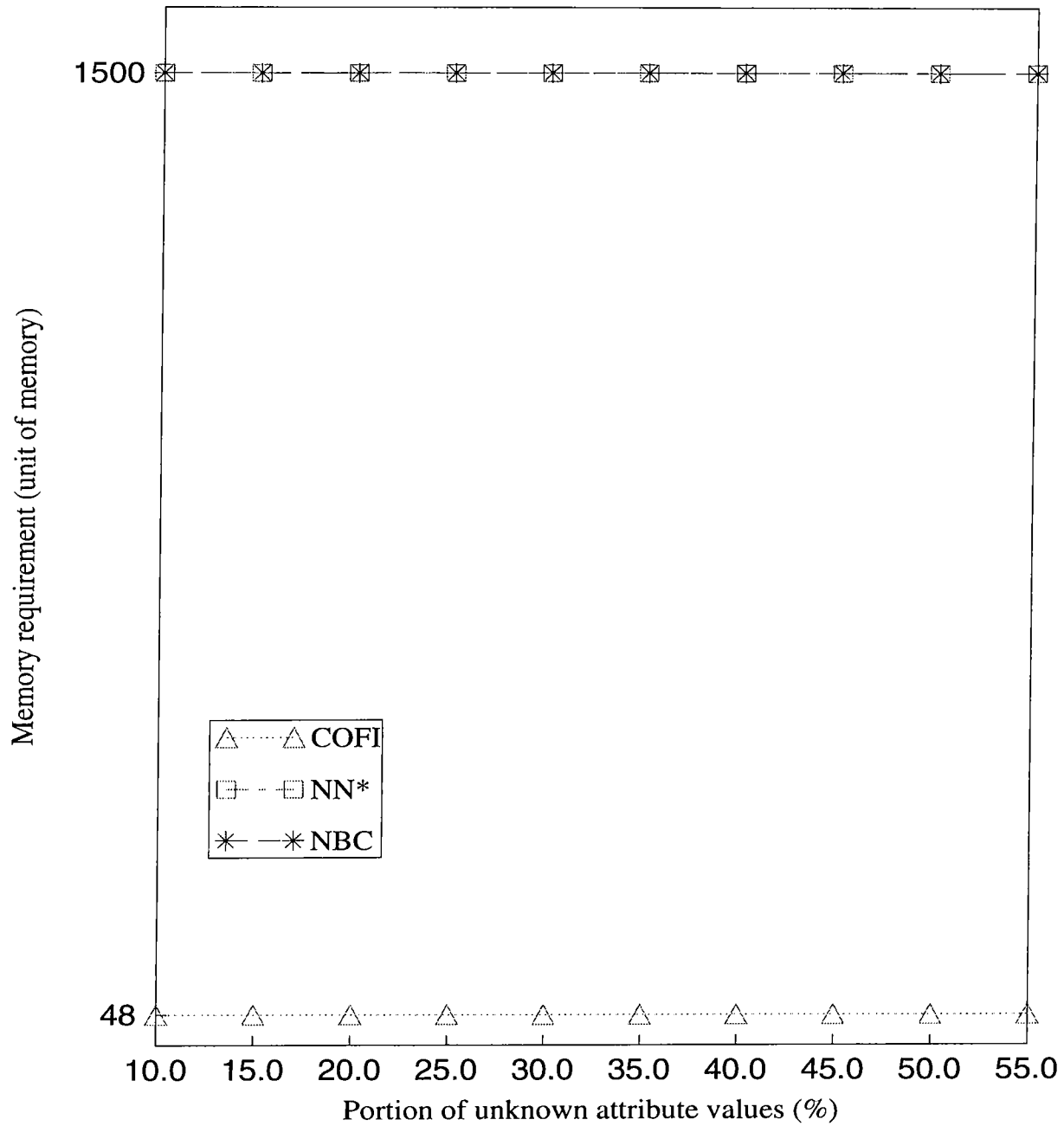


Figure 4.6. Memory requirement of the the COFI, the NN* and the NBC algorithms on domains with unknown attribute values.

The memory requirement of these three algorithms are not affected by the number of unknown feature values, so the memory requirement is constant through the increase in the missing attribute values. However, the memory requirement of the NBC and NN* is much greater than the COFI algorithm because of the nature of the algorithms. This situation is shown in Figure 4.6.

4.2.3 Experiments with Real-World Datasets

The COFI algorithm has been tested on seven different real-world datasets. These real-world datasets are taken from the collection of databases distributed by the machine learning group at the University of California at Irvine. The use of real-world datasets provides a measure for the system's accuracy on noisy and incomplete datasets. In addition to this, it allowed comparisons between COFI and similar machine learning algorithms. Detailed explanations and comparisons will be given about these datasets in the following pages. Here, we have used the leave-one-out technique as a testing methodology.

We have compared the COFI algorithm with the NBC, the NN* and the CFP algorithms. In these tests, the NBC and the NN* algorithms do not require any domain specific parameters. However, several domain dependent parameters should be set before the execution of the CFP and the COFI algorithms.

In the testing of the CFP algorithm, GA-CFP is used to determine the domain dependent parameters of the CFP algorithm, that is, these domain specific parameters have been learned by GA-CFP. GA-CFP has used randomly selected 20% of the dataset as its training data. The population has been selected as 100 for the genetic algorithm. Uniform crossover is used with the crossover probability of $p_c=0.7$. The probability of mutation is $p_m=0.01$. The chromosome with best fitness value in 50 generations is used to set the domain dependent parameters of the CFP algorithm. The fitness value of a chromosome is two-fold cross-validation accuracy of the CFP algorithm on the training dataset randomly selected 20% of the dataset. In the mutation process, an allele is multiplied by a randomly selected value between 0.5 and 1.5. The domain specific parameters of the CFP algorithm are the generalization distances for each feature and the weight adjustment rate, Δ . On the other

Table 4.1. An overview of the real-world datasets.

Dataset	Size	# of Features	# of Linear Features	# of Classes	Unknown Features	Baseline Accuracy
iris	150	4	4	3	0%	33%
glass	214	9	9	6	0%	36%
horse-colic	368	22	7	3	30%	61%
ionosphere	351	34	34	2	0%	64%
hungarian	294	13	6	2	0%	64%
cleveland	303	13	6	2	0%	54%
wine	178	13	13	2	0%	39%

hand, we have only one domain dependent parameter, generalization ratio g in the COFI algorithm and it is externally given to the algorithm. Therefore, it is not easy to set this parameter properly. The disadvantage is the difficulty of selecting the generalization ratio which provides the best interval construction for each feature. However, since we have only one domain dependent parameter, the results of the algorithm are less dependent to the changes on the data than the results of the CFP algorithm.

An overview of the datasets is shown in Table 4.1. In this table, name of the real-world datasets are shown with the size of the dataset, number of features, number of linear features, number of classes, percentage of the unknown attribute values, and the baseline accuracy. The baseline accuracy of a dataset is the accuracy that will be obtained by predicting the class of any test instance as the class of the most frequently occurring class.

In the following pages, brief explanations will be given about the seven different real-world datasets that are tested by the NBC, the NN*, the CFP and the COFI algorithms.

Iris Flowers: Iris flowers dataset from Fisher [19] consists of four integer valued continuous features and a particular species of iris flower. There are three different classes: *iris virginica*, *iris setosa*, *iris versicolor*. The four attributes measured were sepal length, sepal width, petal length and petal width. The dataset contains 150 instances, 50 instances of each three classes. The accuracy of the COFI algorithm obtained for generalization ratio $g = 0.020$ is 95.33%, and with this accuracy result, it achieves better than the NBC and the

Table 4.2. Required memory and execution time of training for the COFI Algorithm.

Dataset	Average Number of Intervals	Time (msec.)
iris	137.75	13.348
glass	314.83	42.187
horse-colic	923.02	90.182
ionosphere	70.01	85.517
hungarian	48.99	51.090
cleveland	69.99	52.817
wine	39.00	16.980

NN* algorithms. Its performance is only 0.67% less than the CFP algorithm. Table 4.2 shows the memory requirements of the COFI algorithm in terms of the number of the intervals and the time of learning phase for the COFI algorithm. Table 4.3 shows the accuracy results of the NBC, NN*, CFP and COFI algorithms.

Glass Data: This dataset consists of attributes of glass samples taken from the scan of an accident. The glass dataset contains 214 instances of which belongs to one of six classes. In this dataset there are 9 features. All feature values are continuous. In the processing of this dataset, the generalization ratio is given as 0.022. Results are shown in Table 4.3. Here, the CFP and the NBC algorithms achieves better accuracies than the COFI algorithm. The performance difference between the COFI and the CFP is again very little, 1.87%. The NBC algorithm performs the best accuracy for this dataset, and the performance difference between the COFI and the NBC algorithm is 2.34%.

Horse-Colic Data: In this dataset there are 368 instances. Number of attributes is 22 and the number of classes is 3. Seven of these features are linear and fifteen of them are nominal. The 30% of the feature values is missing (unknown). We have set the generalization ratio as 0.020 for this dataset. Table 4.3 presents the accuracy results.

Ionosphere Data: The radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. *Good* radar returns are those

Table 4.3. Accuracy results (%) of the algorithms for real-world datasets using leave-one-out evaluation technique.

Dataset	NBC	NN*	CFP	COFI
iris	93.33	92.67	96.00	95.33 (g=0.020)
glass	57.01	45.33	56.54	54.67 (g=0.022)
horse-colic	67.30	61.30	61.14	67.66 (g=0.020)
ionosphere	79.77	88.03	88.60	94.30 (g=0.270)
hungarian	79.59	63.95	81.29	84.69 (g=0.280)
cleveland	80.53	80.20	85.48	81.85 (g=0.079)
wine	93.26	78.65	91.01	94.94 (g=0.043)

showing evidence of some type of structure in the ionosphere. *Bad* returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. Instances in this database are described by 2 attributes per pulse number, corresponding to the complex values returned by the function resulting from the complex electromagnetic signal. David Aha briefly investigated this database. He found that k-nearest neighbor attains an accuracy of 92%, that Ross Quinlan's C4 algorithm attains 94.0% (with no windowing).

In this data there are 351 data instances and 34 continuous attributes. The number of classes is 2, that is, this is a classification problem. There are no missing values in this dataset. Here the generalization ratio is 0.270 and achieved accuracy is shown in Table 4.3.

Hungarian and Cleveland Data: Both datasets are about the heart disease diagnosis. Each dataset has the same instance format. Cleveland data was collected from the Cleveland Clinic Foundation and Hungarian data was collected from the Hungarian Institute of Cardiology.

These databases contain 76 attributes originally, but in machine learning field 13 of them is used. All attributes are numeric valued and 6 of them have nominal values. The class is determined according to the presence of heart disease, that is, this is binary classification problem. There are no missing values in these datasets for the features that we have used. Accuracy results and the memory requirements are shown in Table 4.3 and Table 4.4 respectively.

Table 4.4. Average memory requirements of the algorithms for real-world datasets.

Dataset	NBC	NN*	CFP	COFI
iris	750	750	594.36	551.00 ($g=0.020$)
glass	2140	2140	5389.12	1259.32 ($g=0.022$)
horse-colic	6900	6900	5309.80	3692.08 ($g=0.090$)
ionosphere	12285	12285	16893.52	280.04 ($g=0.270$)
hungarian	4116	4116	1863.64	195.96 ($g=0.280$)
cleveland	4242	4242	2430.84	279.96 ($g=0.079$)
wine	2492	2492	5197.48	156.00 ($g=0.430$)

Wine Data: This dataset is about recognizing wine types. This data is provided by Pharmaceutical and Food analysis and technologies. The classes are separable. In a classification context, this is a well-posed problem with “well behaved” class structures. This dataset is the result of the chemical analysis of wines grown in the same region in Italy but derived from three different cultures. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The dataset contains 178 instances. The class distribution of the data is given in Appendix. All attribute values are continuous. The generalization ratio is 0.043. Table 4.3 shows the results.

It is seen that, the COFI algorithm always achieves better results than the NN* algorithm. The COFI algorithm gives the best accuracy result for the horse-colic, ionosphere, hungarian and wine datasets among these four algorithms. The NBC algorithm reaches better accuracy result than the COFI algorithm for only the glass dataset. However, the performance difference between the the COFI algorithm and the NBC algorithm is only 2.34%. The CFP and the COFI algorithm is usually achieves similar accuracy results. The COFI algorithm gives better accuracy results for the horse-colic, the ionosphere, the hungarian, and the wine datasets than for the CFP algorithm. For the iris dataset CFP is only 0.67% worse than the CFP algorithm, for the glass dataset this difference is 1.87%, and for the cleveland dataset CFP achieves 3.63% better accuracy than the COFI algorithm. It is seen that if the overall performance is considered, the COFI algorithm achieves the best accuracy results among these four algorithms for the seven datasets.

If memory requirements are compared among these algorithms, it will be seen that, the COFI algorithm always uses less memory than the other algorithms. The intervals are stored in memory in the COFI algorithm, and the number of intervals depends on the generalization ratio and the structure of the dataset, in other words, the distribution of the feature values. Four parameters are kept in memory for each interval, these elements are lower and upper bounds of the interval, representativeness count and the associated class value for the COFI algorithm. In the NBC and NN* algorithms, all the instances should be kept in memory verbatim. Therefore, for each instance the feature values and the associated class value should be stored in memory. The table 4.4 give the memory requirements of these algorithms for the seven real-world datasets. Here we assumed that one unit of memory is allocated for each element of an interval and for each feature and class value. Since the execution time depends on the search among these stored information in the memory, the COFI algorithm is faster than the NN* and the NBC algorithms.

In the CFP algorithm, segments are stored in memory. However, since no overlapping is allowed and subpartitioning is done in the CFP algorithm, usually number of segments of the CFP algorithm is greater than the number of intervals of the COFI algorithm. Memory requirement of the CFP algorithm is also shown in Table 4.4.

Chapter 5

Conclusions and Future Work

In this thesis, we have presented a new methodology of learning from examples which is a new form of exemplar-based generalization technique based on the representation of overlapping feature intervals. This technique is called as Classification with Overlapping Feature Intervals (COFI).

COFI is an inductive, incremental and supervised concept learning method. It learns the projections of the intervals in each class dimension for each feature. Those intervals correspond to the learned concepts.

When compared to Naive Bayesian Classifier (NBC) and Nearest Neighbor (NN) algorithms, COFI is similar to them, in that all of these techniques process each feature separately. All of them use feature based representation. The classification process is based on majority voting for these algorithms. However, COFI requires much less memory than others, because in the training process, NBC and NN keep all examples with all feature and class values in memory separately. However, in COFI, only intervals are stored. Therefore, when compared to many other similar techniques, the COFI algorithm's memory requirement is less than their requirements.

The COFI algorithm is applicable to domains, where each feature is independent from others. Number of features and number of classes are not important for the COFI algorithm. Both nominal and linear feature values can be processed successfully.

An important characteristic of the COFI algorithm is its ability of overlapping. When compared to CFP, COFI is successful when CFP is failed in some cases. For example, if the projection of concepts on an axis are overlapping each other, the CFP algorithm constructs many segments of different classes next to each other. In that case, the accuracy of classification depends on the observed frequency of the concepts. However, in the COFI algorithm, since all class dimensions are independent from each other, no specialization is required. The concept descriptions can be overlapped.

Another important property of the COFI algorithm is its way of handling the unknown attribute values. Most of the systems use *ad hoc* methods to handle the unknown attribute values [23, 44]. Like CFP, the COFI algorithm also ignores the unknown attribute values. Since the value of each attribute is handled separately, this causes no problem.

The behavior of the COFI algorithm to the irrelevant features is very interesting. Irrelevant attributes can easily be detected by looking at the concept description of the COFI algorithm. Irrelevant attributes constructs intervals that cover whole dimension for each class. Therefore, the detection of the irrelevant attributes can be performed by looking at the intervals of the COFI algorithm. In the CFP algorithm, irrelevant attributes cause many number of fragmentations on the feature dimensions. While the COFI algorithm decreases the number of intervals, the CFP algorithm increases its segments during the processing of irrelevant attributes.

The COFI algorithm uses relative representativeness count for prediction. Using relative representativeness count provides a kind of normalization, because datasets usually contain non-homogeneous examples in terms of the number of examples that have the same class value. In the COFI algorithm, intervals store four kinds of information; it's lower and upper bound, associated class value and representativeness count. Representativeness count is the number of examples that interval represents. This number is divided to the total number of examples which have the same class value. Therefore, a kind of normalization is achieved to make the correct predictions.

One important component of the COFI algorithm is the generalization ratio g . It controls the generalization process. This ratio is chosen externally depending on experiments. For a future work, an algorithm may be developed

to find the optimum generalization ratio or genetic algorithms may be used to find the optimum generalization ratio.

One of the most important property of the COFI algorithm is that, one may easily predict the class of a given test instance by using the learned concepts. The algorithm does not have to search all features. However, in some techniques, for example in decision trees, the algorithm has to search all features until it reaches to a leaf. In the COFI algorithm, a decision can be made by just looking at some key attributes. This approach is similar to the human approach of classification.

At the end, it should be expressed the simplicity of the rules for the concept descriptions in the COFI algorithm. The simplicity of the algorithm does not affect the accuracy results when compared to the very complex algorithm NBC. NBC represent its knowledge in the form of probability distribution functions. Simple-rule learning systems are generally a viable alternative to systems that learn more complex rules by applying more complex algorithms. If a complex rule is induced, then it's additional complexity must be justified by giving more accurate predictions than a simple rule.

When compared to the NBC and the NN* algorithms, the COFI algorithm uses much less storage, because both the NBC and the NN* algorithms should keep all feature values separately in the memory to find the probability density function in NBC and to find the distance metric in NN* for predictions. In the COFI algorithm, intervals should be kept in memory. The memory required for intervals is usually much less than the required memory for the NBC and the NN* algorithms.

As a final word, we should repeat that the knowledge representation scheme in the form of overlapping feature intervals is a viable technique in classification. The COFI algorithm can compete with the well-known machine learning algorithms in terms of accuracy. Also the requirement of much less memory and the high execution speed of the algorithm are the important advantages of the algorithm. This algorithm may be applicable to other problems encountered in artificial intelligence.

Bibliography

- [1] D.W. Aha, Incremental, Instance-Based Learning of Independent and Graded Concept Descriptions, In *Proceedings of the Sixth International Workshop Machine Learning*, pp: 387-391, Ithaca, NY: Morgan Kaufmann, 1989.
- [2] D.W. Aha, Tolerating Noisy, Irrelevant and Novel Attributes in Instance-Based Learning Algorithms, *International Journal of Man-Machine Studies*, 36:267-287, 1992.
- [3] D.W. Aha and D. Kibler, Noise-Tolerant Instance-Based Learning Algorithms, In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp: 794-799, Detroit, MI: Morgan Kauffman, 1989.
- [4] D.W. Aha, D. Kibler and M.K. Albert, Instance-Based Learning Algorithms, *Machine Learning*, 6:37-66, 1991.
- [5] M.K. Albert and D.W. Aha, Analyses of Instance-Based Learning Algorithms, In *Proceedings of the Ninth National Conference on Artificial Intelligence* pp: 553-558, 1991.
- [6] H. Almuallim and T.G. Dietterich, Learning with Many Irrelevant Features, In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp: 547-552, 1991.
- [7] D. Angluin and P. Laird, Learning from Noisy Examples, *Machine Learning*, 2:343-370, 1988.
- [8] J.G. Carbonell, editor, *Machine Learning: Paradigms and Methods*, The MIT Press, 1990.

- [9] P. Clark and T. Niblett, Induction in Noisy Domains, In I. Bratko and N.Lavrac (Eds.), *Progress in Machine Learning*, pp:11-30, Wilmslow, England:Sigma Press, 1987.
- [10] T.M. Cover and P.E. Hart, Nearest Neighbor Pattern Classification, *IEEE Transactions on Information Theory*: 13:21-27, 1967.
- [11] G. Dejong and R. Mooney, Explanation-Based Learning : An Alternative View, *Machine Learning*, 1:145-176, 1986.
- [12] G. Dejong, Learning with Genetic Algorithms: An Overview, *Machine Learning*, 3:121-128, 1988.
- [13] P.J. Denning, The Science of Computing, *American Scientist*, volume: 77, pp: 216-219, 1989.
- [14] R.D. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.
- [15] B. Efron, *The Jackknife, the Bootstrap and Other Resampling Plans*, In SIAM, Philadelphia, Pa., 1982.
- [16] U.M. Fayyad and K.B. Irani, Technical Note, On The handling of Continuous-Valued Attributes in Decision Tree Generation, *Machine Learning*, 8:87-102, 1992.
- [17] U.M. Fayyad and K.B. Irani, Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, In *Proceedings of the Thirteenth International Joint Conference on AI*, pp: 1319-1324, Morgan Kaufmann, 1993.
- [18] E.A. Feigenbaum, The Simulation of Verbal Learning Behavior, In E.A. Feigenbaum and J. Feldman (Eds.), *Computers and Thoughts*, New York, McGraw-Hill, Inc., 1963.
- [19] D.H. Fisher, Knowledge Acquisition Via Incremental Conceptual Clustering, *Machine Learning*, 2:139-172, 1987.
- [20] J.M. Fitzpatrick and J.J. Grefenstette, Genetic Algorithms in Noisy Environments, *Machine Learning*, 3:101-120, 1988.

- [21] J.A. Freeman and D.M. Skapura, *Neural Networks Algorithms, Applications and Programming Techniques*, Addison-Wesley Publishing Company, 1991.
- [22] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, San Diego, 1990.
- [23] J.W. Grzymala-Busse, On the Unknown Attribute Values in Learning from Examples, In *Proceedings of Sixth International Symposium Methodologies for Intelligent Systems*, pp: 368-377, October 1991.
- [24] H.A. Güvenir and İ. Şirin, A Genetic Algorithm for Classification by Feature Partitioning, In *Proceedings of the fifth International Conference on Genetic Algorithms*, pp: 543-548, 1993.
- [25] H.A. Güvenir and İ. Şirin, Complexity of the CFP, a Method for Classification Based on Feature Partitioning, In *Proceedings of the Third Congress of the Italian Association for Artificial Intelligence, AI*AI'93*, Advances in Artificial Intelligence, LNAI 728, Pietro Torasso (Ed.), Torino, Italy, October 1993.
- [26] R.C. Holte, Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, *Machine Learning*, 11:63-91, 1993.
- [27] E. Hunt, J. Marin and P. Stone, *Experiments in Induction*, New York, Academic Press, 1966.
- [28] L. Kanal and Chandrasekaran, On Dimensionality and Sample Size In Statistical Pattern Classification, *Pattern Recognition*, pp: 225-234, 1971.
- [29] J.D. Kelly and L. Davis, A Hybrid Genetic Algorithm for Classification, In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp: 645-650, 1991.
- [30] P. Langley, Toward a Unified Science of Machine Learning, *machine Learning*, 3:253-259, 1989.
- [31] P. Lachenbruch and M. Mickey, Estimation of Error Rates in Discriminant Analysis, *Technometrics*, 1-111, 1968.
- [32] R.P. Lippmann, An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, April 1987.

- [33] N. Littlestone, Learning Quickly When Irrelevant Attributes Abound: A New Linear-Threshold Algorithm, *Machine Learning*, 1:47-80, 1986.
- [34] H. Lounis and G. Bisson, Evaluation of Learning Systems: An Artificial Data-Based Approach, In *Proceedings of European Working Session on Learning*, pp: 463-481, 1991.
- [35] D. Medin and M. Schaffer, Context Theory of Classification Learning, *Psychological Review*, 85:3, 207-238, 1978.
- [36] R.S. Michalski and R.L. Chilausky, Learning by Being Told and Learning from Examples: An Experimental Comparison of the Two methods of Knowledge Acquisition In the Context of Developing an Expert System for Soybean Disease Diagnosis, *International Journal of policy Analysis and Information Systems*, 4, 1980.
- [37] S.N. Minton, Selectively Generalizing Plans for Problem Solving, *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp: 596-599, Los Angeles, CA: Morgan Kaufmann, 1985.
- [38] T.M. Mitchell, An Analysis of Generalization As a Search Problem, *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, vol:1, pp: 577-582, 1979.
- [39] T.M. Mitchell, R. Keller and S. Kedar-Cabelli, Explanation-Based Generalization: A Unifying View, *Machine Learning*, 1:47-80, 1986.
- [40] B. Müller and J. Reinhardt, *Neural Networks An Introduction*, Springer-Verlag Berlin Heidelberg, 1990.
- [41] J.R. Quinlan, Discovering Rules From Large Collections of Examples: A Case Study, In D. Michie (Ed.), *Expert Systems in the Microelectronic Age*, Edinburgh: edinburgh University Press, 1979.
- [42] J.R. Quinlan, Learning Efficient Classification Procedures and Their Application to Chess and Games, In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), *Machine Learning, An Artificial Approach*, Los Altos: Morgan Kaufmann, 1983.
- [43] J.R. Quinlan, Induction of Decision Trees, *Machine Learning*, 1:81-106, 1986.

- [44] J.R. Quinlan, Decision Trees as Probabilistic Classifiers, In *Proceedings of Fourth International Workshop on Machine Learning*, pp: 31-37, June 1987.
- [45] J.R. Quinlan, Unknown Attribute Values in Induction, In A. Segre (Ed.), *Proceedings of the 16th International Workshop on Machine Learning*, pp: 164-168, San Mateo, CA:Morgan Kaufmann, 1989.
- [46] J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, California, 1993.
- [47] J. Rachlin, S. Kasif, S. Salzberg and D.W. Aha, Towards a Better Understanding of Memory-Based Reasoning Systems, *International Machine Learning Conference*, 1994.
- [48] L. Rendell, A General Framework for Induction and a Study of Selective Induction. *Machine Learning*, 1:177-226, 1986.
- [49] S. Salzberg, *Learning with Generalized Exemplars*, Kluwer Academic Publishers, Massachusetts, 1990.
- [50] S. Salzberg, A Nearest Hyperrectangle Learning Method, *Machine Learning*, 6:251-276, 1991.
- [51] G. Stanfill and D. Waltz, Toward Memory-Based Reasoning, *Communications of the ACM* 29:1213-1228, 1986.
- [52] J.C. Schlimmer and R.H. Granger, Incremental Learning from Noisy Data, *Machine Learning*, 1:317-354, 1986.
- [53] J. Shavlik, R.J. Mooney and G. Towell, Symbolic and Neural Learning Algorithms: An Experimental Comparison, *Machine Learning*, 6: 111-143, 1991.
- [54] R.S. Sutton, Learning to Predict by the Methods of Temporal Differences, *Machine Learning*, 3:9-44, 1988.
- [55] A. Szladow and W. Ziarko, Rough Sets: Working with Imperfect Data, *AI Expert*, pp: 36-42, July 1993.
- [56] İ. Şirin and H.A. Güvenir, A Classification Algorithm Based on Feature Partitioning, In *Proceedings of the Second TAINN Symposium*, p: 283-288, 1993.

- [57] İ. Şirin and H.A. Güvenir, Empirical Evaluation of the CFP Algorithm, In *Proceedings of the Sixth Australian Joint Conference on Artificial Intelligence*, pages 311-315, 1993.
- [58] İ. Şirin and H.A. Güvenir, *An Algorithm for Classification by Feature Partitioning*, Technical Report CIS-9301, Bilkent University, Dept.of Computer Engineering and Information Science, Ankara, 1993.
- [59] A. Thayse (ed.), *From NLP to Logic for Expert System, A Logic Based Approach to AI*, pp: 263-356, 1991.
- [60] K.M Ting and R.M. Cameron-Jones, Exploring a Framework For Instance Based Learning and Naive Bayesian Classifiers, em *Proceedings of the 7th Australian Joint Conference on Artificial Intelligence*,Armidale, Austarlia, 21-25 November, 1994.
- [61] H.G. Ünsal and H.A. Güvenir, Soil Classification by Using CFP, In *Proceedings of the Third TAINN Symposium*, pp: 257-265, 1994.
- [62] S.M. Weiss and I. Kapouleas, An Empirical Comparison of Pattern Recognition, Neural Nets, and Machine Learning Classification Methods,*Proceedings of the Eleventh International Joint conference on Artificial Intelligence*, San Mateo, CA:Morgan Kaufmann, 1990.
- [63] P.H. Winston, *Artificial Intelligence*, Addison-Wesley, Reading-Massachusetts, 1984.

Appendix A

Concept Descriptions of the Real-World Datasets

In this section, we will give the concept descriptions learned by the COFI algorithm for all of the real-world datasets that are used in this thesis. Here, the accuracy is determined as the average of the randomized runs.

In the following figures, in the first and the top box, the name of the example dataset, the training ratio t , the generalization ratio g , the number of the example that is being processed, the exact class value of the current example and feature values of the example are shown during the training phase. In the testing phase, the predicted class value, number of the correct predictions are added to this top box. At the end, that is, when the algorithm finishes its work, number of run, associated accuracy result and the average of the runs until the last one are also printed in this box.

In the second and the larger box, the construction of the intervals can be followed during the execution of the algorithm. While class intervals are being formed, current generalization distance, minimum and maximum feature values up to current example and the property of the feature, that is, whether it is linear or non-linear, are printed. After the execution, final rectangles represents the concept descriptions. In the following pages, concept descriptions of five real-world datasets are presented.

The concept description differs as the order of the training instances changes.

The accuracy depends on the order of the training instances, that is, the COFI algorithm has an incremental structure. For each randomized run a different accuracy is obtained, and in the following figures the concept description of the randomized run which gave the maximum accuracy among the 50 runs is presented.

Here, one important characteristic of the COFI algorithm is seen on the ionosphere dataset. The concept description of this dataset shows that all class dimensions are wholly covered by only one interval for some features. From Figure A.9 to Figure A.12, the concept description of this dataset is presented, and irrelevant features are easily detected in these figures by just looking at the structure of the intervals. For example, f_7 is one of the irrelevant features, because, for all class dimensions there exist only one interval that cover whole class dimension.

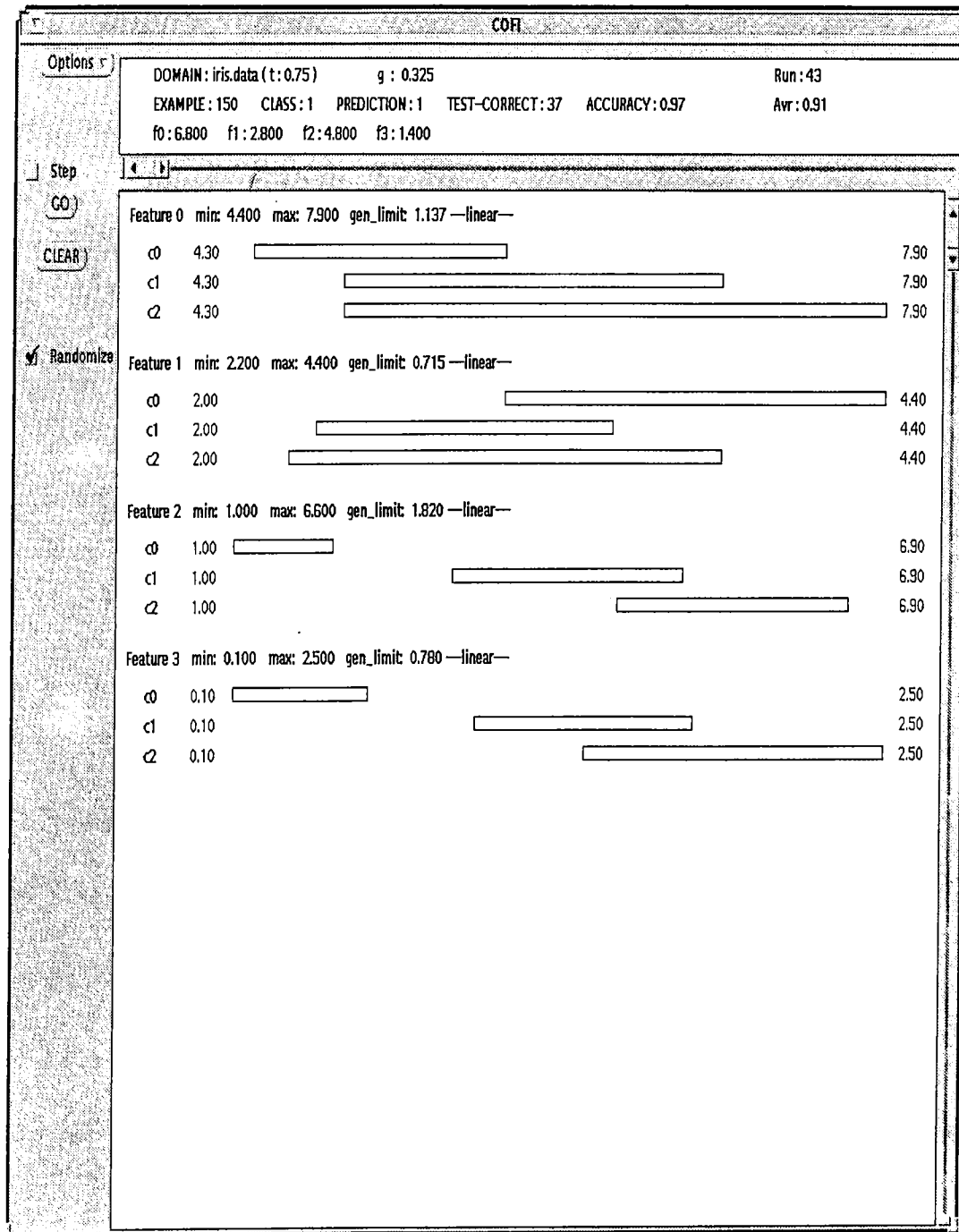


Figure A.1. Concept description of the iris dataset.

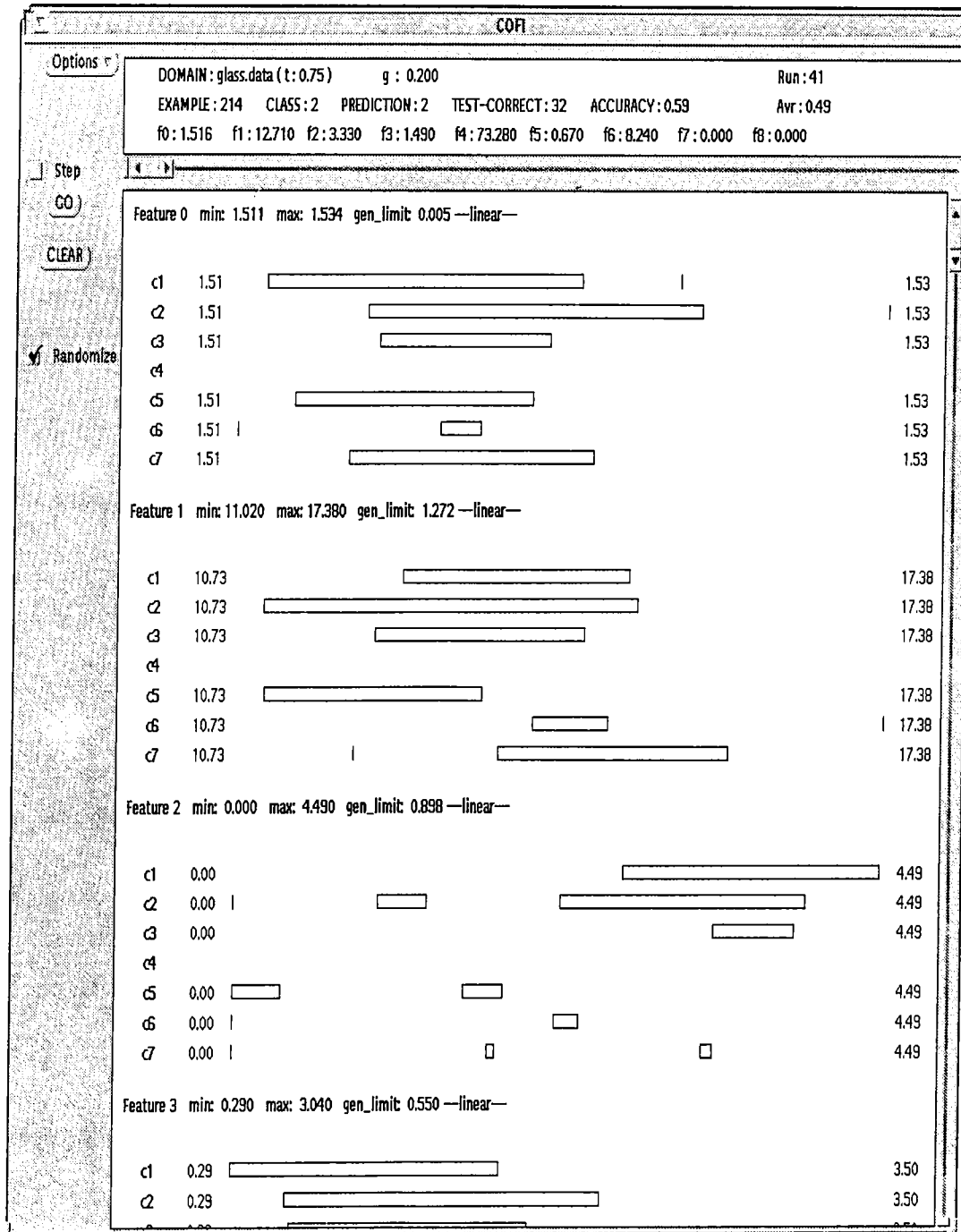


Figure A.2. Concept Description of the glass dataset.

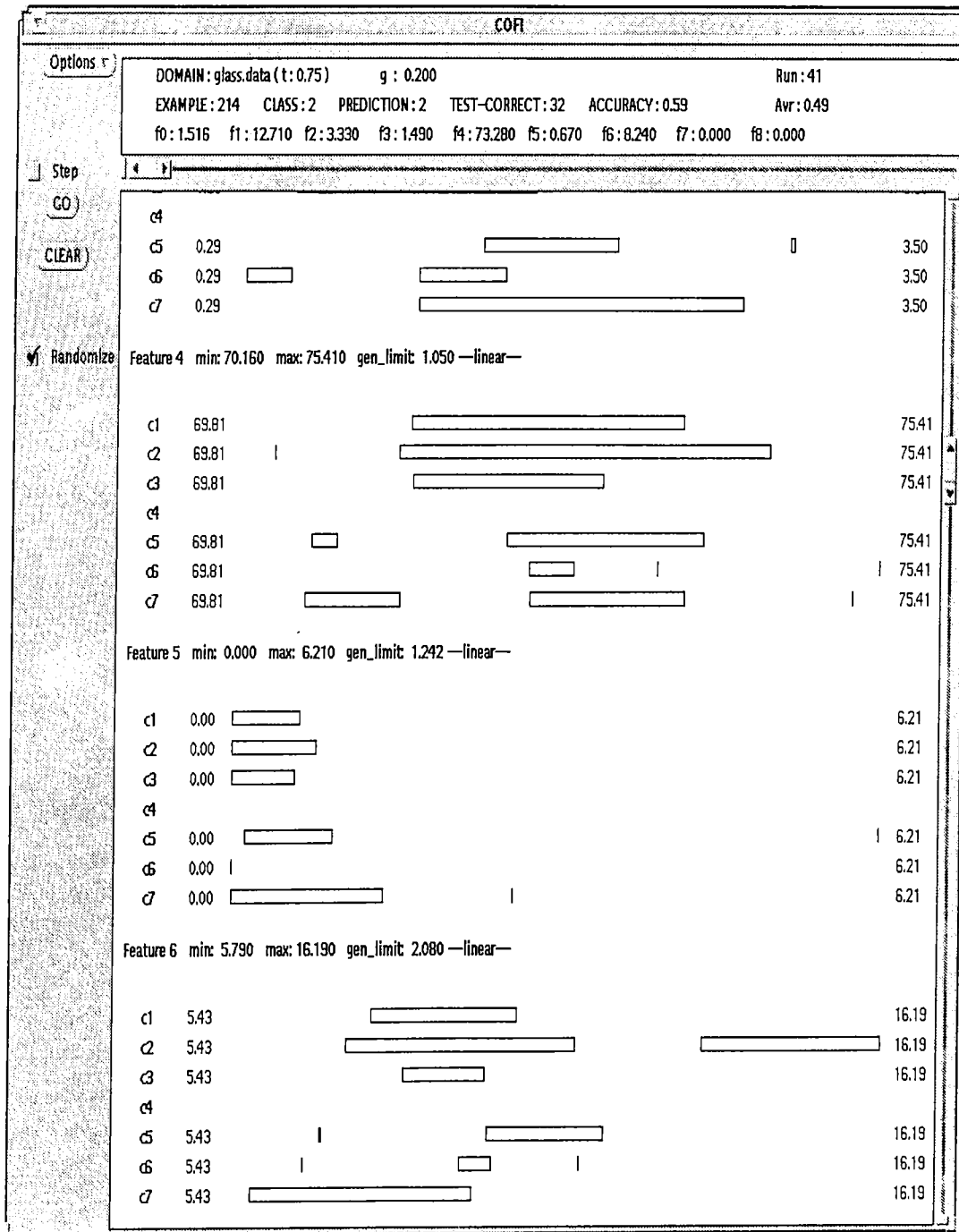


Figure A.3. Concept Description of the glass dataset - continued.

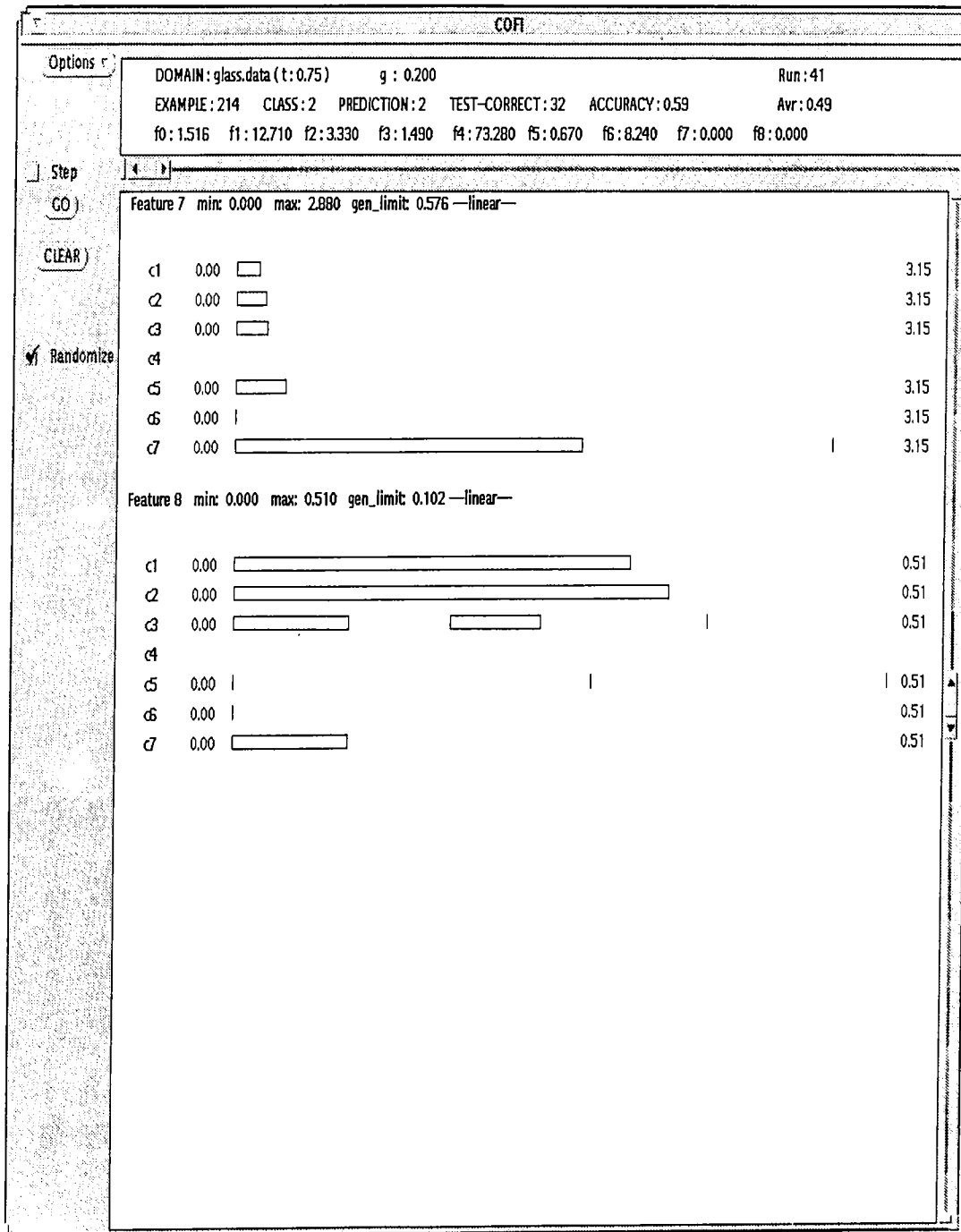


Figure A.4. Concept Description of the glass dataset - continued.

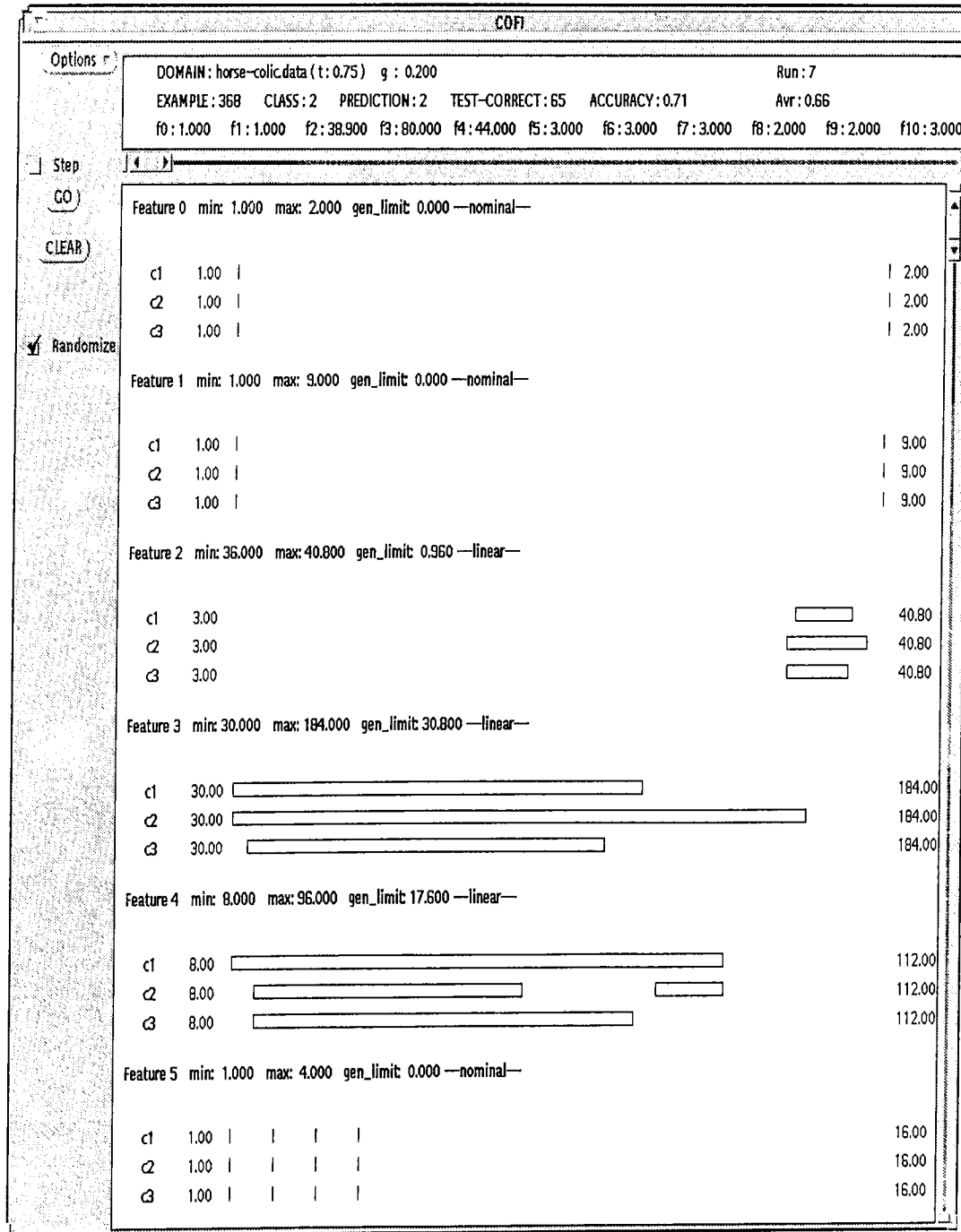


Figure A.5. Concept description of the horse-colic dataset.

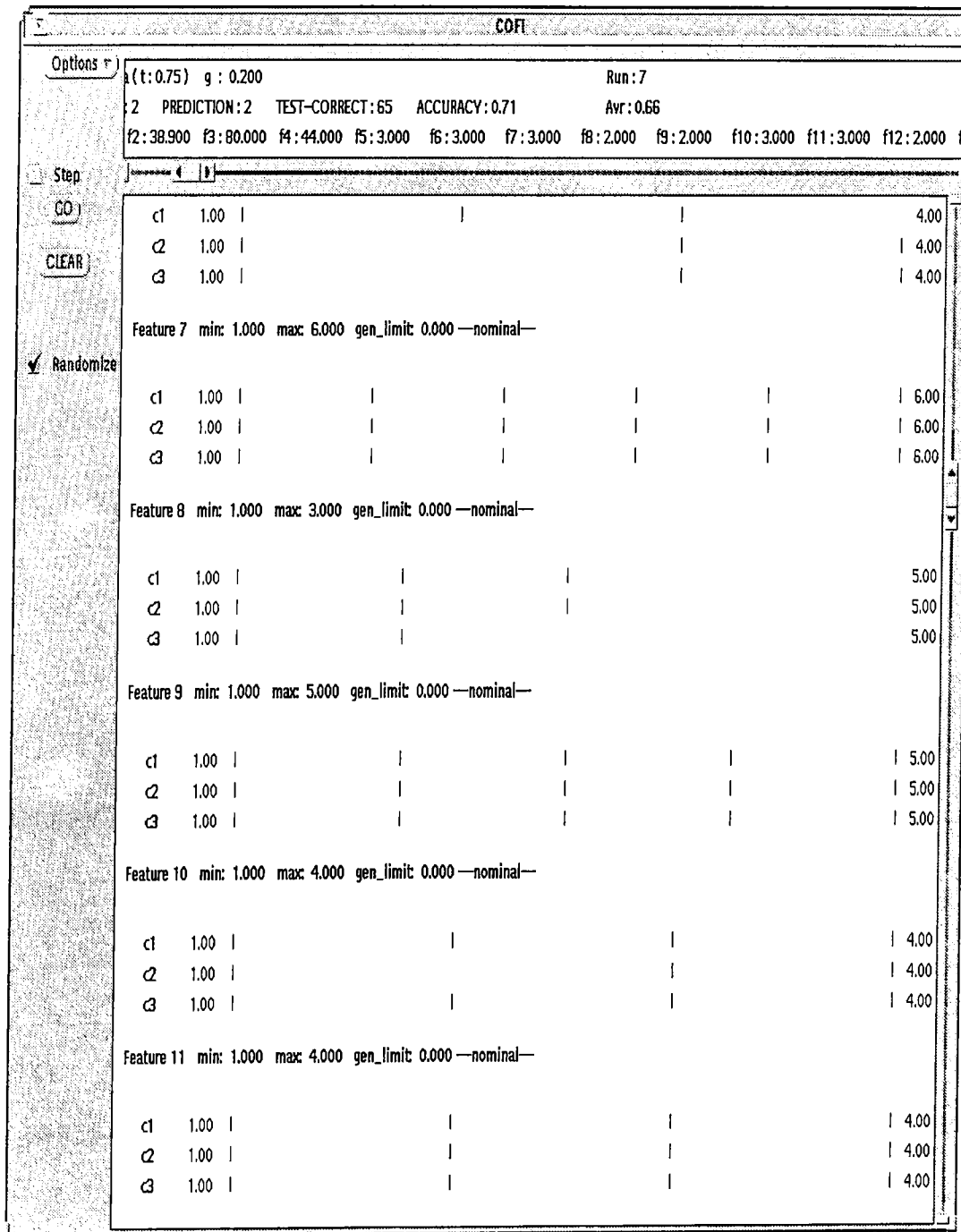


Figure A.6. Concept description of the horse-colic dataset - continued.

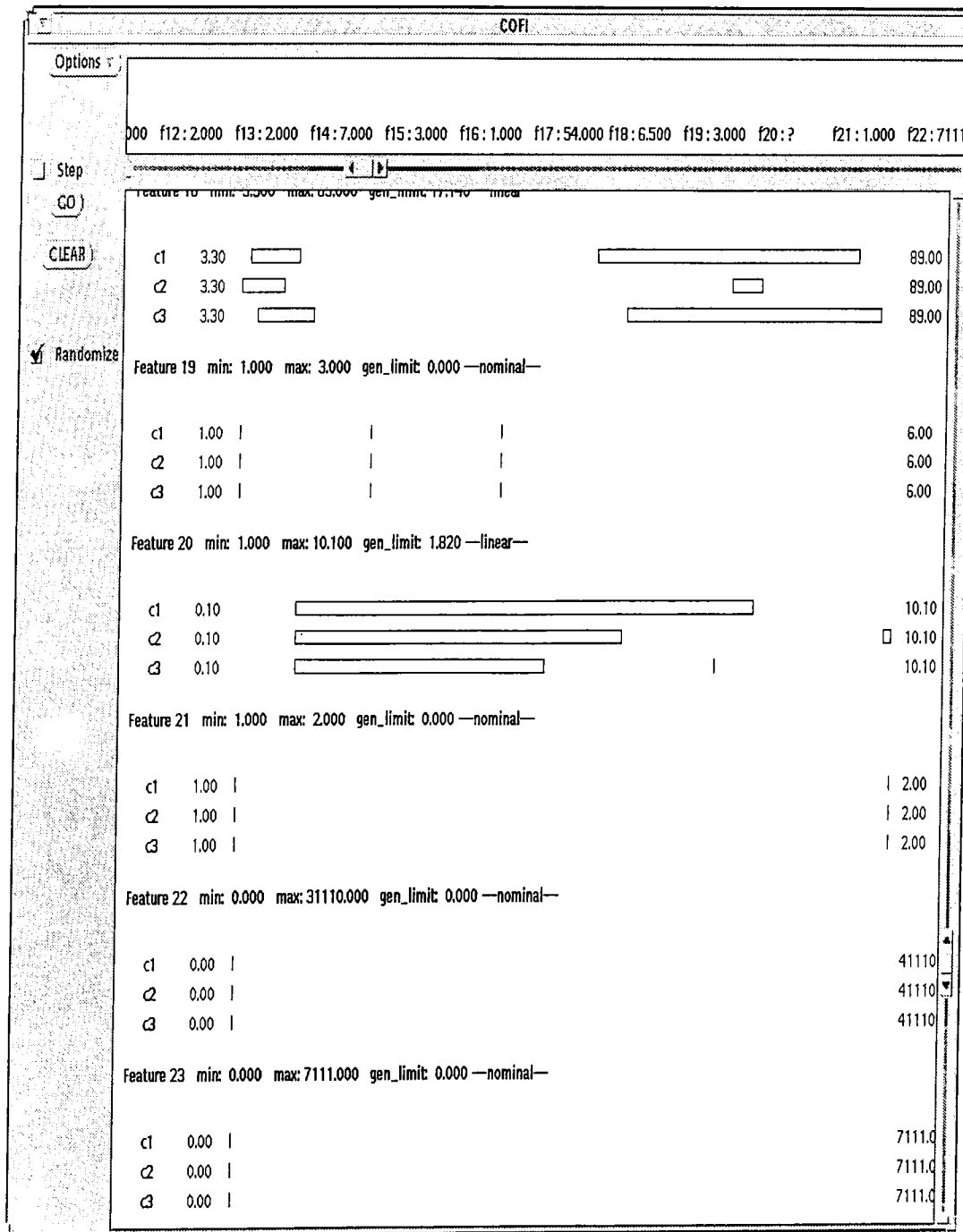


Figure A.7. Concept description of the horse-colic dataset - continued.

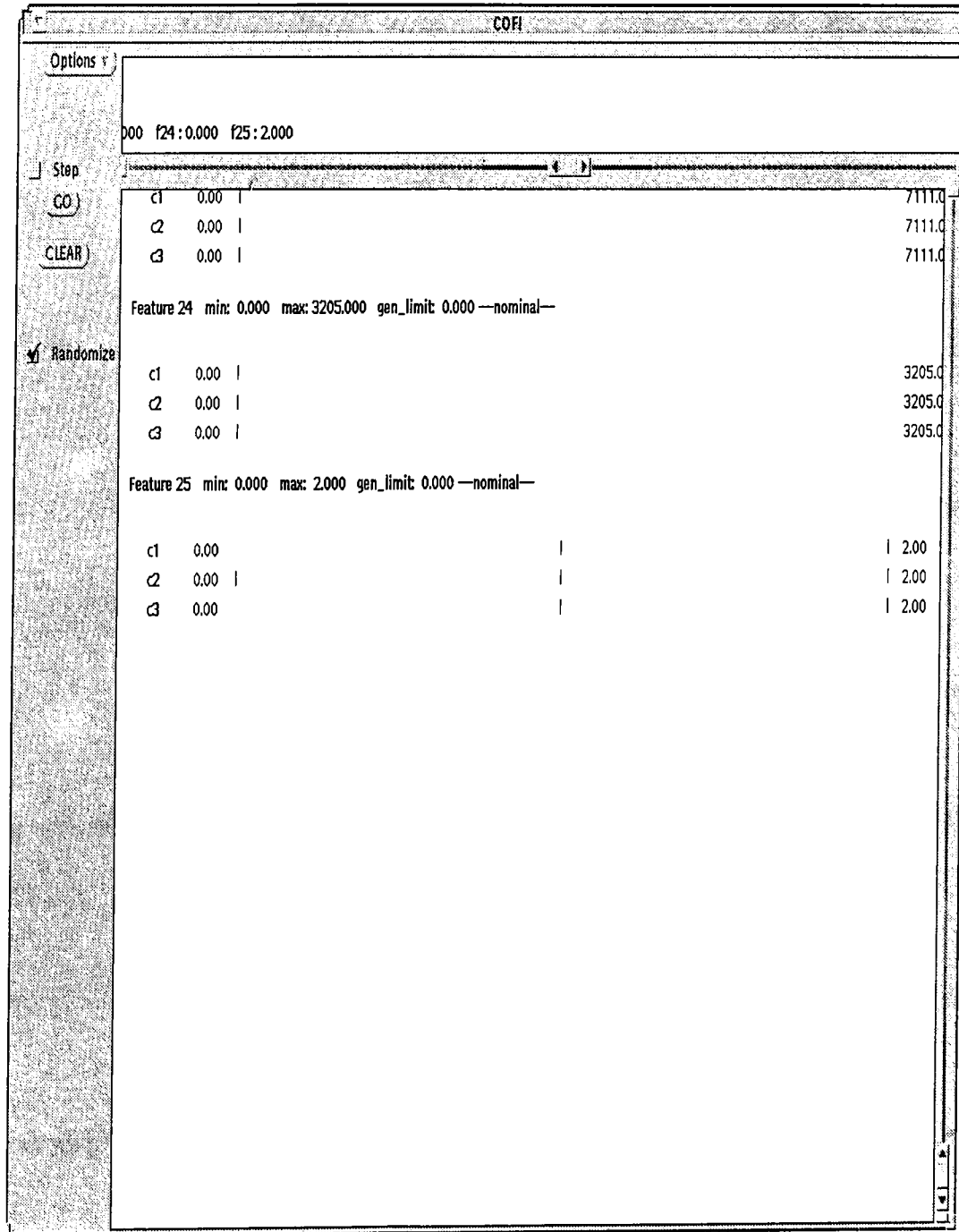


Figure A.8. Concept description of the horse-colic dataset - continued.

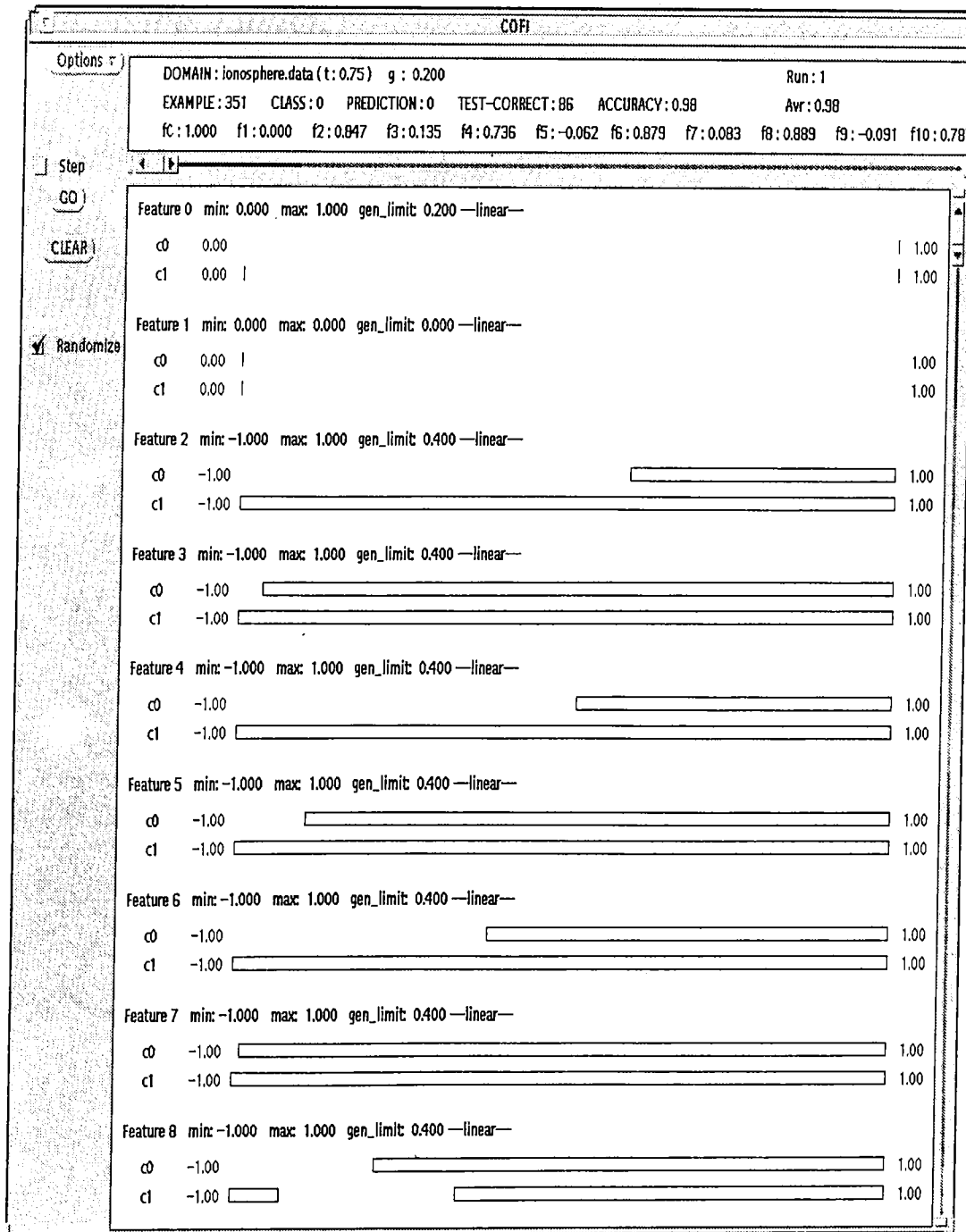


Figure A.9. Concept description of the ionosphere dataset.

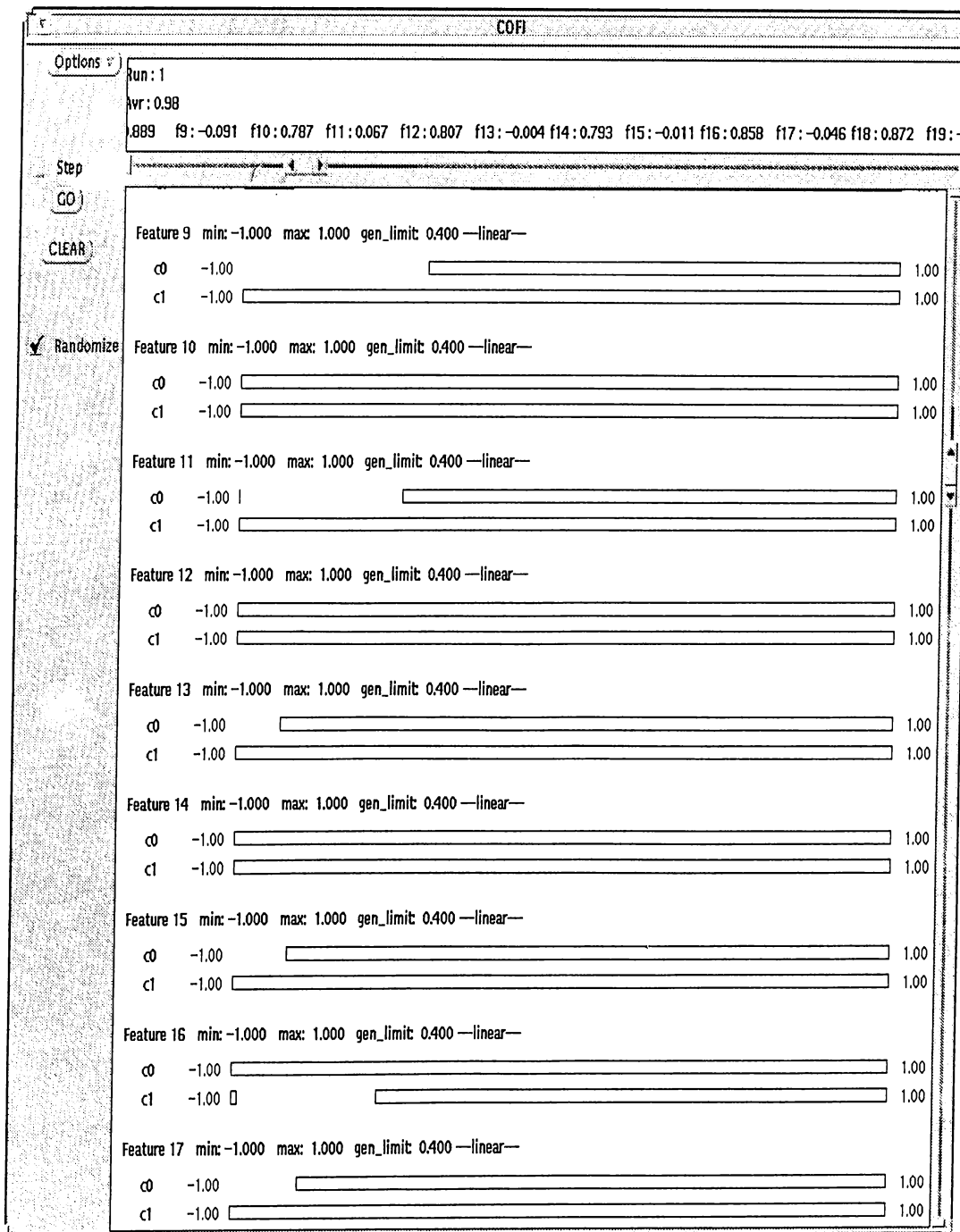


Figure A.10. Concept Description of the ionosphere dataset - continued.

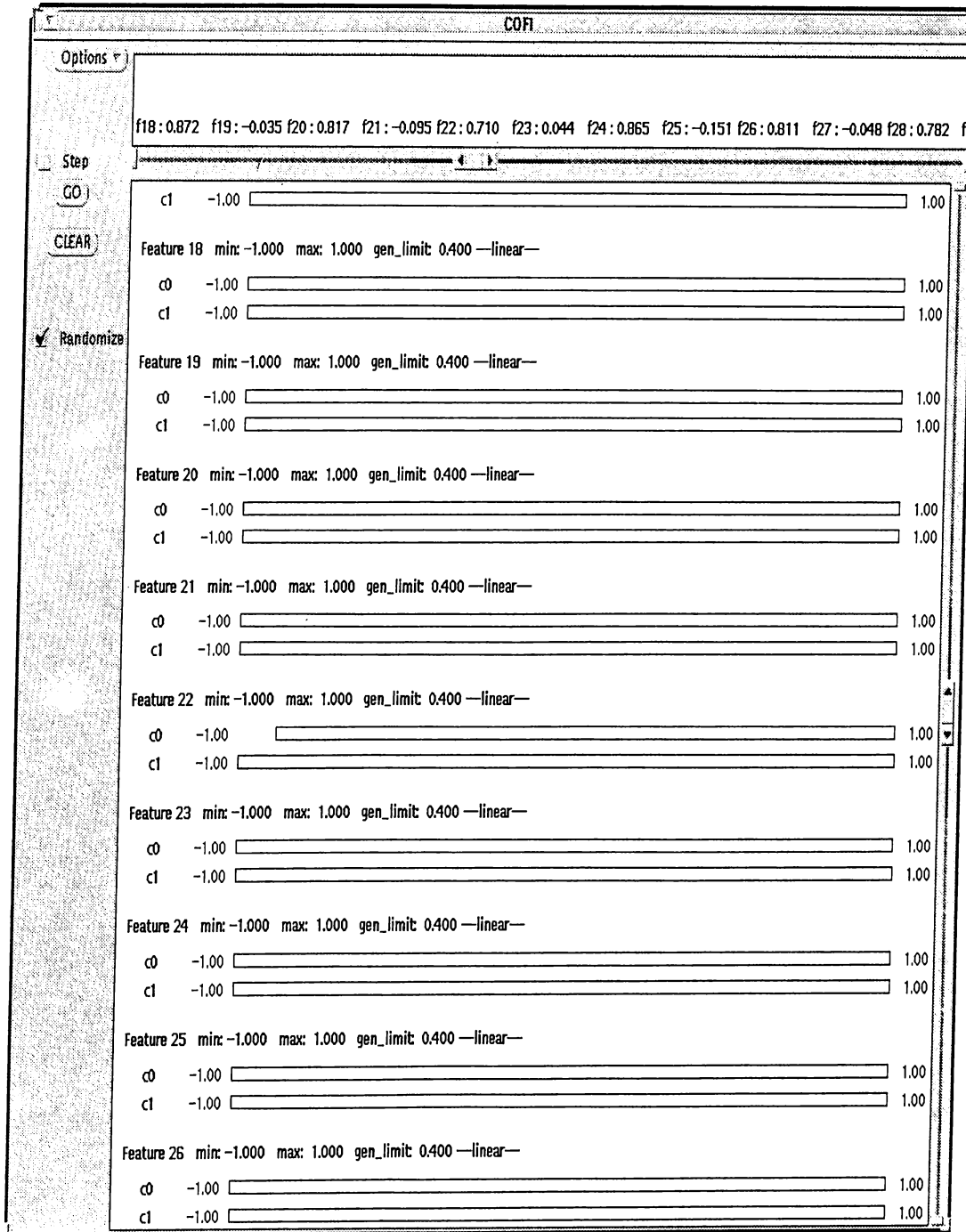


Figure A.11. Concept description of the ionosphere dataset - continued.

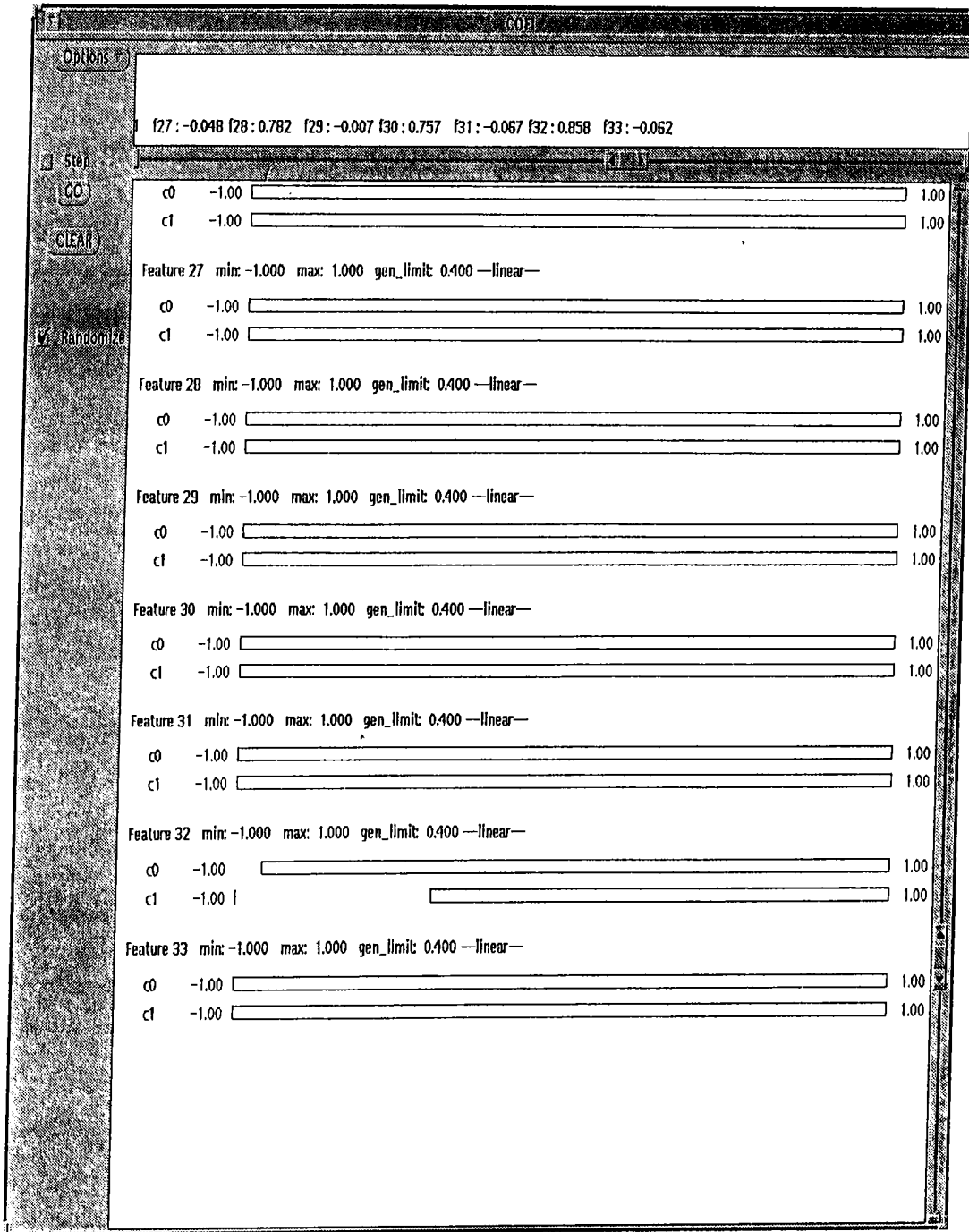


Figure A.12. Concept description of the ionosphere dataset - continued.

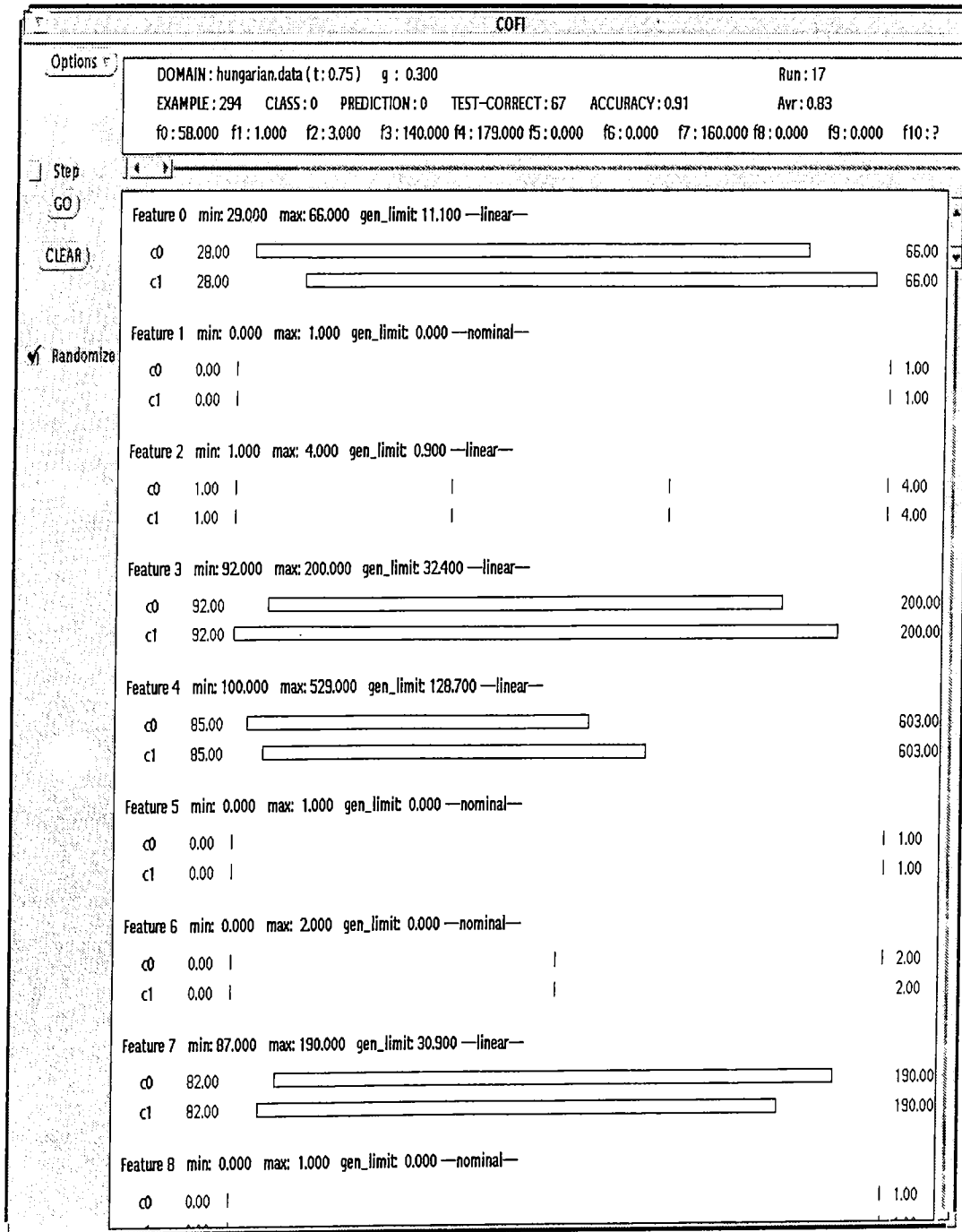


Figure A.13. Concept description of the hungarian dataset.

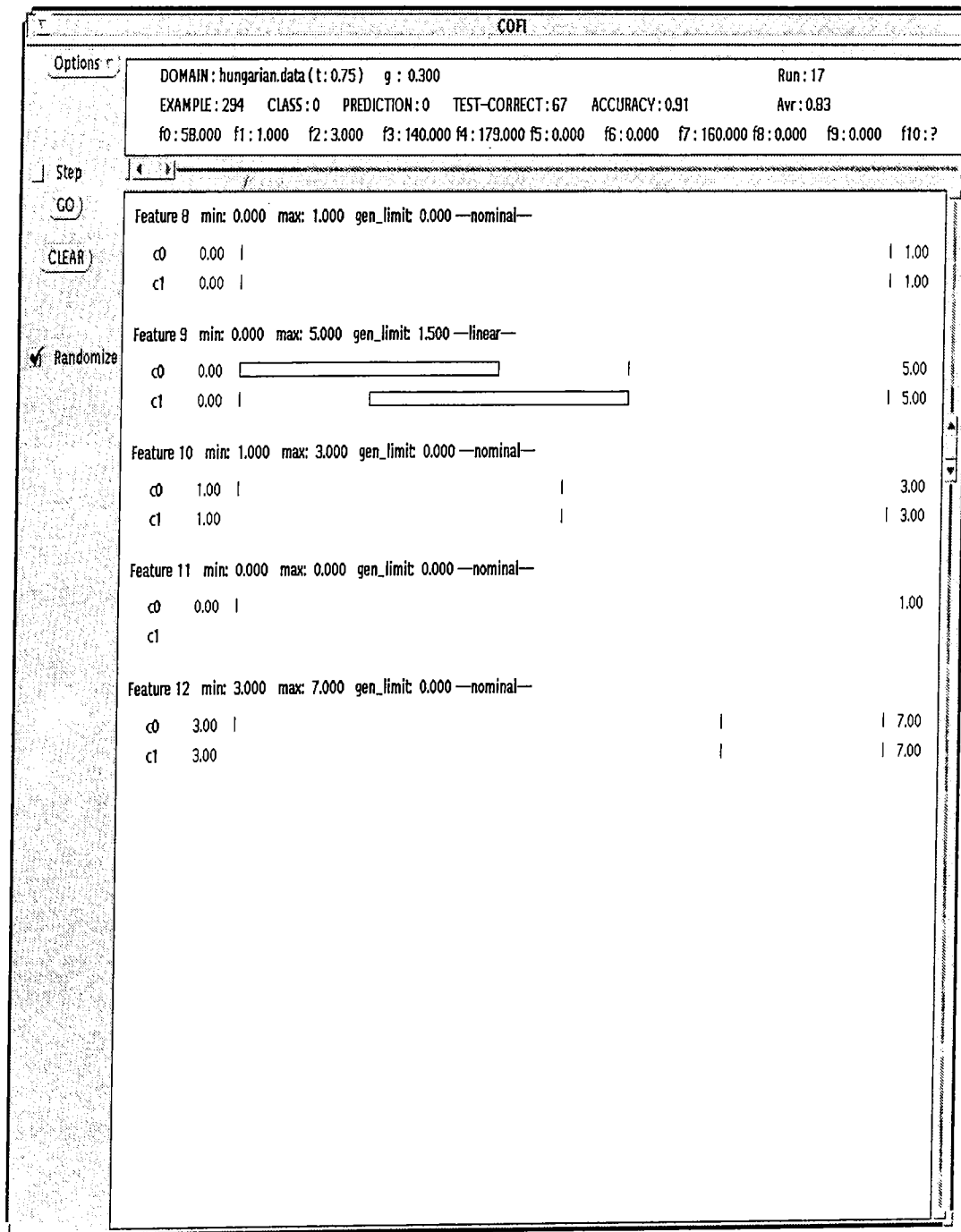


Figure A.14. Concept description of the hungarian dataset - continued.

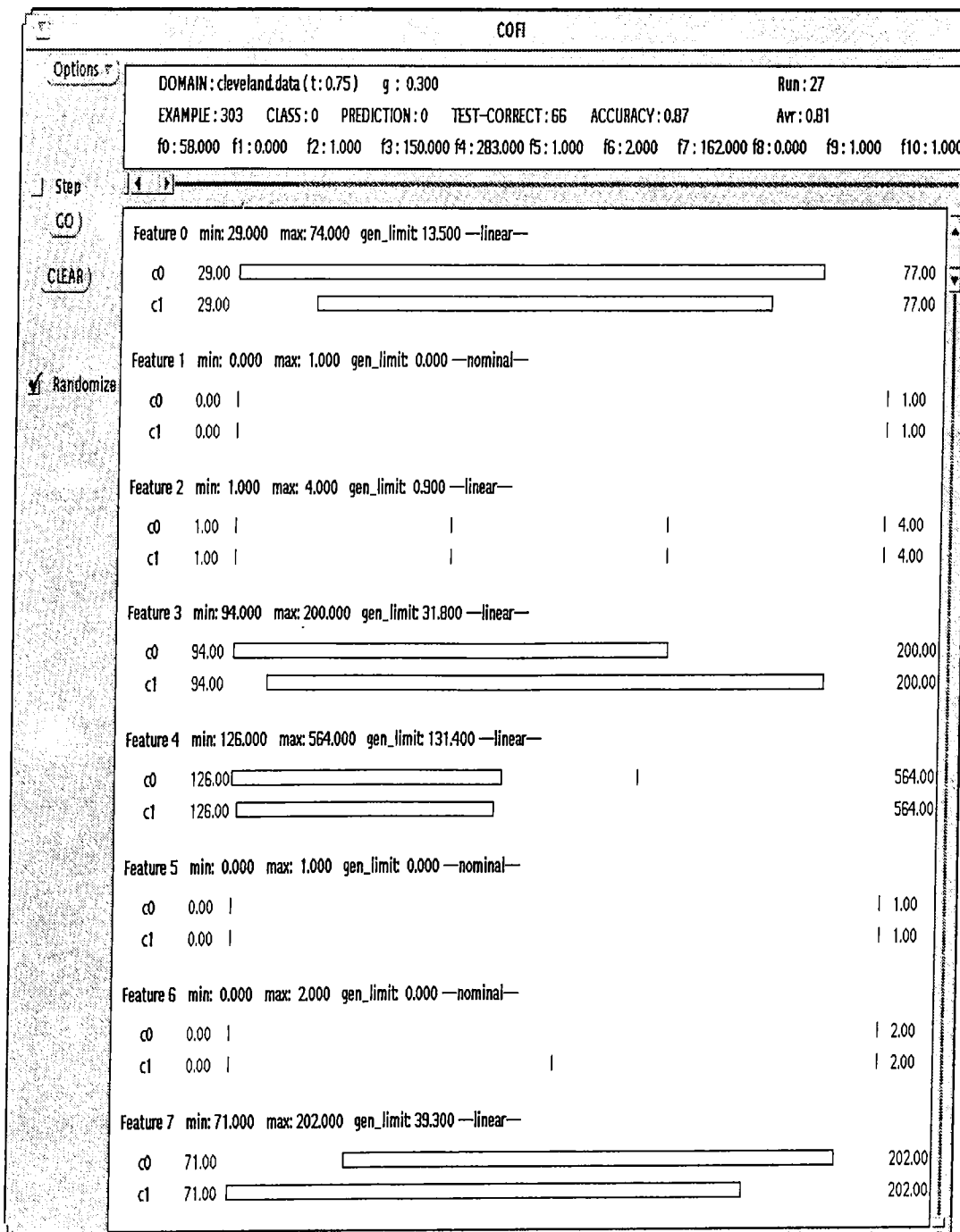


Figure A.15. Concept description of the cleveland dataset.

COFI

Options ▾

DOMAIN: cleveland.data (t: 0.75) g: 0.300 Run: 27
 EXAMPLE: 303 CLASS: 0 PREDICTION: 0 TEST-CORRECT: 66 ACCURACY: 0.87 Avr: 0.81
 f0: 58.000 f1: 0.000 f2: 1.000 f3: 150.000 f4: 283.000 f5: 1.000 f6: 2.000 f7: 162.000 f8: 0.000 f9: 1.000 f10: 1.000

Step | 1 |

GO

CLEAR

Randomize

Feature 8 min: 0.000 max: 1.000 gen_limit: 0.000 —nominal—

c0	0.00				1.00
c1	0.00				1.00

Feature 9 min: 0.000 max: 6.200 gen_limit: 1.850 —linear—

c0	0.00		<div style="border: 1px solid black; width: 100%; height: 15px;"></div>		6.20
c1	0.00		<div style="border: 1px solid black; width: 100%; height: 15px;"></div>		6.20

Feature 10 min: 1.000 max: 3.000 gen_limit: 0.000 —nominal—

c0	1.00				3.00
c1	1.00				3.00

Feature 11 min: 0.000 max: 3.000 gen_limit: 0.000 —nominal—

c0	0.00				3.00
c1	0.00				3.00

Feature 12 min: 3.000 max: 7.000 gen_limit: 0.000 —nominal—

c0	3.00				7.00
c1	3.00				7.00

Figure A.16. Concept description of the cleveland dataset - continued.

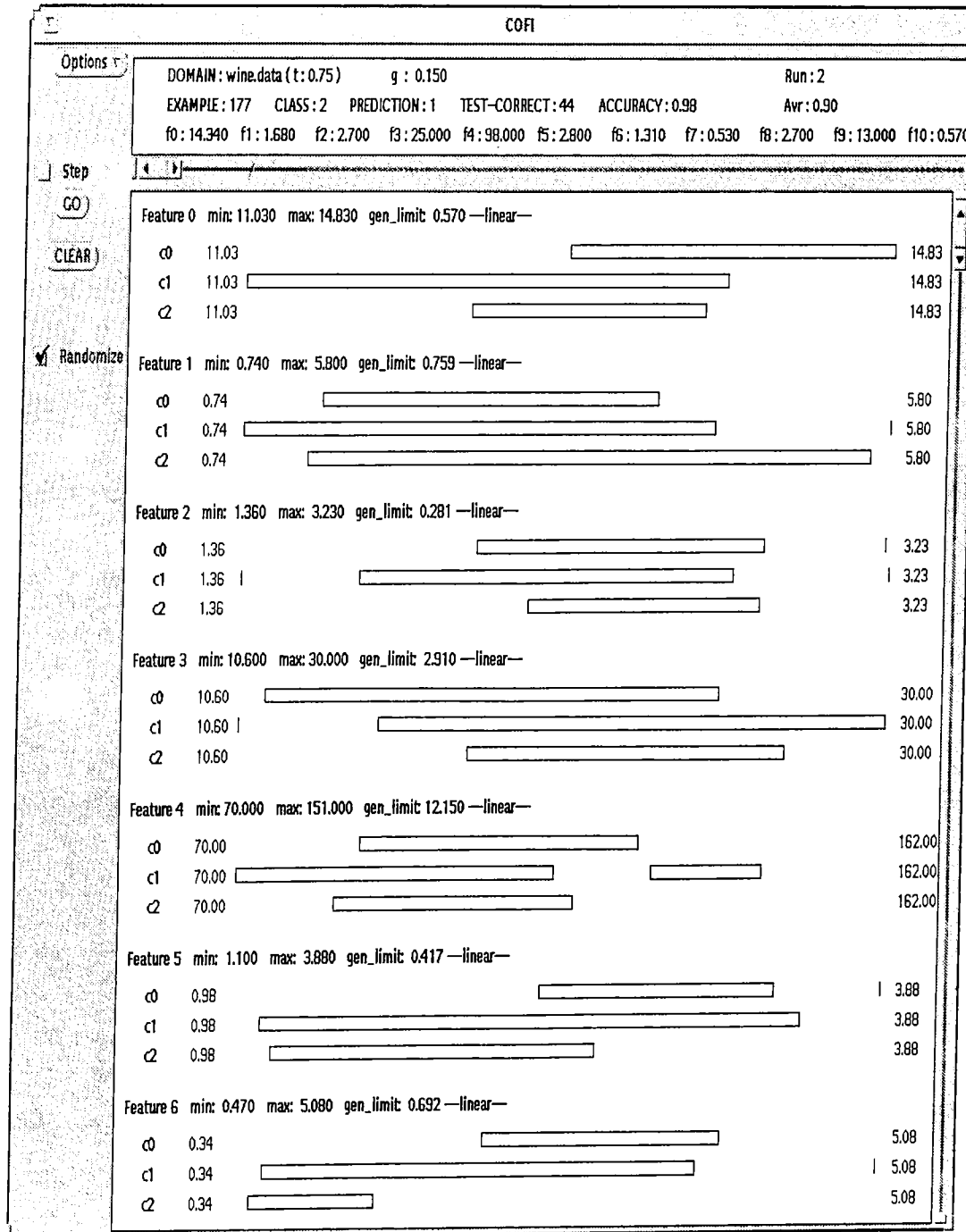


Figure A.17. Concept description of the wine dataset.

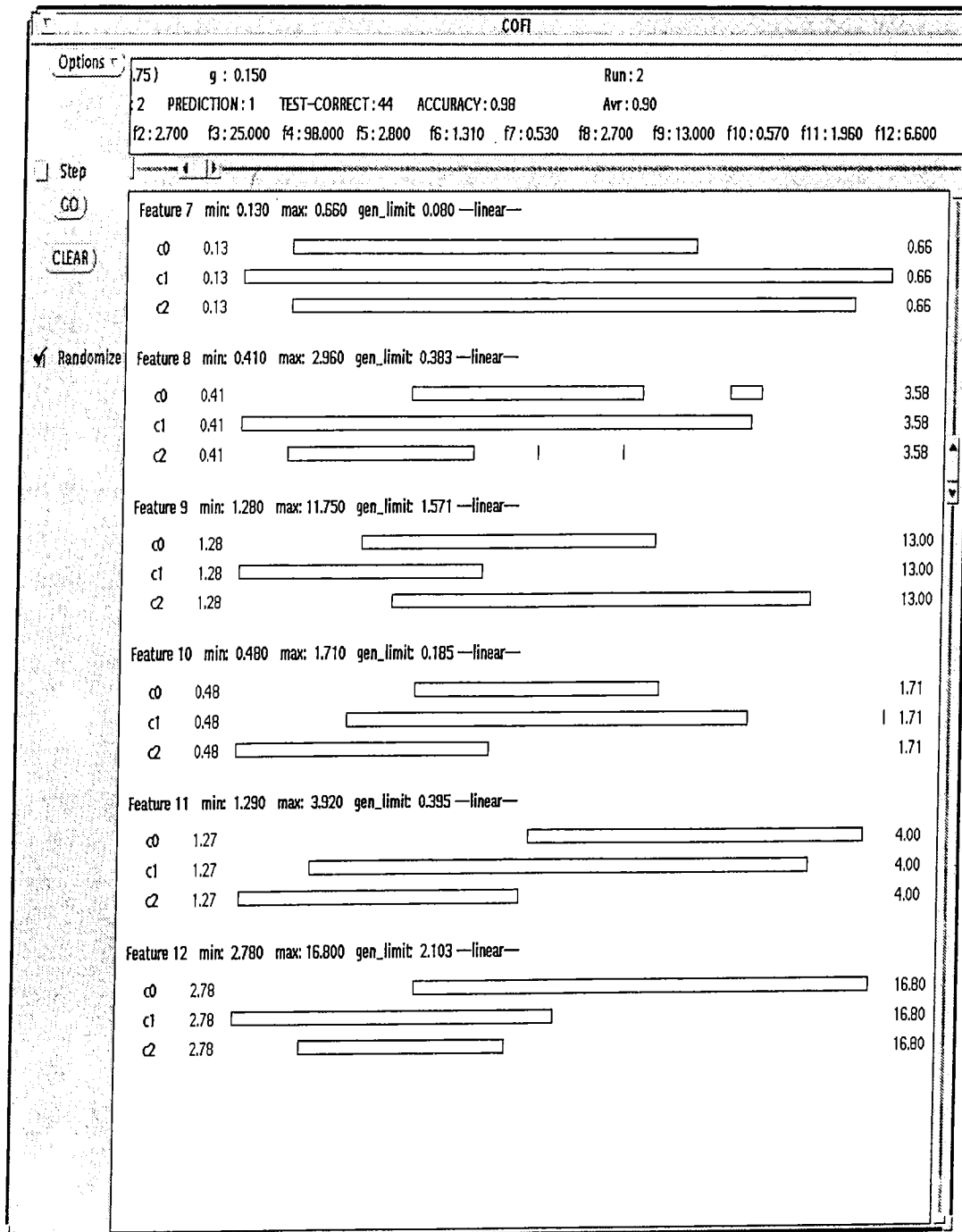


Figure A.18. Concept description of the wine dataset - continued.