CONCEPT REPRESENTATION WITH OVERLAPPING FEATURE INTERVALS

H. ALTAY GUVENIR
Department of Computer Engineering and Information Science
Bilkent University, 06533 Ankara, Turkey

HAKIME G. KOÇ
Aselsan, 06172 Ankara, Turkey

Abbreviated title: Overlapping Feature Intervals

Mailing Address:
Prof. H. Altay Guvenir
Department of Computer Engineering and Information Science
Bilkent University, 06533 Ankara, Turkey

Abstract

This article presents a new form of exemplar-based learning method, based on overlapping feature intervals. In this model, a concept is represented by a collection of overlapping intervals for each feature and class. Classification with Overlapping Feature Intervals (COFI) is a 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 feature dimension for each class. Initially, an interval is a point on a feature-class dimension, then it can be expanded through generalization. No specialization of intervals is done on feature-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. An evaluation of COFI and its comparison with similar other classification techniques is given.

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 in any form of learning.

One of the central insights of AI is that intelligence involves search, and that effective search is constrained by domain specific knowledge. 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.

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 (Rendell, 1986).

Inducing a general concept description from examples and counterexamples is one of the most widely studied methods for symbolic learning. The goal is to develop a description of a concept from which all previous positive instances can be re-derived while none of the previous negative instances can be re-derived by the same process of re-derivation 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 paper, we propose a new symbolic model for concept learning, based on the representation of *overlapping feature intervals* (OFI). The Classification with Overlapping Feature Intervals (COFI, for short) algorithm is a particular implementation of this technique. Feature intervals are formed by generalizing the feature values of training instances from the same class. Overlapping concept descriptions are allowed, that is there may exist different classes for the same feature values. However, 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* of the interval. The intervals are constructed separately for each feature dimension and for each class, called feature-class dimension. The construction is through an incremental generalization from the set of training instances. Initially, an interval is a point, that is, its lower and upper bounds are equal to initial feature values of the first training 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* proposed by Winston (1984). Therefore, the description of a concept (class) is the collection of intervals formed on each feature for that class.

Generalization is the main process of training in the COFI algorithm, since it does not use any specialization heuristic. In order to avoid over-generalization of intervals, generalization is limited with the use of a user specified parameter. Generalization of an interval to include a new training instance depends on that single external variable, called *generalization ratio* and the maximum and the minimum feature value up to current training 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 large number of small intervals to be constructed, whereas large generalization ratios cause small number of large intervals.

Classification in COFI is simply a search for the intervals corresponding to the value of test instance for each feature. A feature votes for the classes of such intervals. The vote of a feature for a class is the *relative representativeness count*, which is the ratio of the representativeness count of the matched interval to the total number of training instances of that class. The votes of a feature are maintained in a *vote vector*, whose elements represent the votes for each

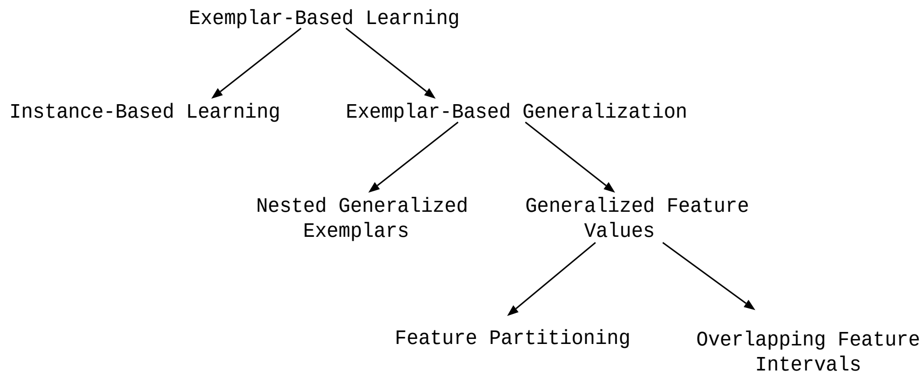

Figure 1: Exemplar-based learning algorithms.

class. For the final classification of a test instance, the vote vectors of features are summed. The class that received the highest amount of votes is declared to be the class of the test instance.

The COFI algorithm handles unknown attribute and class values in a straight forward manner. 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 (Quinlan, 1993).

In the next section, we will present some of the existing concept learning models. Then the detailed explanation about our new algorithm, COFI, will be given. Later, we will evaluate the COFI algorithm by giving the complexity analysis and the results of empirical evaluation on some real-world datasets.

EXEMPLAR-BASED MODELS

Exemplar-based learning is a kind of concept learning methodology in which the concept definition is constructed from the examples themselves, using the same representation language. There are two main types of exemplar-based learning methodologies in the literature, namely instance-based learning and exemplar-based generalization (see Figure 1). Instance-based learning retains examples in memory as points in feature space and never changes their representation. However, in exemplar-based generalization techniques the point-storage model is slightly modified to support generalization.

An instance-based concept description includes a set of stored instances along with some information concerning their past performance during the training process. The similarity and classification functions determine how the set of saved instances in the concept description are used to predict values for the category attribute. Therefore, IBL concept descriptions contain these two functions along with the set of stored instances.

The instance-based learning technique (Aha et al., 1991) was first implemented in three different algorithms, namely IB1, IB2, and IB3. IB1 stores all the training instances, IB2 stores only the instances for which the prediction was wrong. Neither IB1 nor IB2 remove any instance from concept description after it had been stored. IB3 is the noise tolerant version of the IB2 algorithm. It employs a significance test to determine which instances are good classifiers and which ones are believed to be noisy. Later Aha (1992) implemented two extensions to these algorithms, called IB4, and IB5. IB4 learns a separate set of attribute weights for each concept, which are then used in the similarity function. IB5, which is an extension of IB4, can cope with novel attributes by updating an attribute's weight only when its value is known for both the instance being classified and the instance chosen to classify it.

IBL algorithms assume that instances which have high similarity values according to the similarity function have similar classifications. This leads to their local bias for classifying novel instances according to their most similar neighbor's classification. They also assume that without prior knowledge attributes will have equal relevance for classification decisions (i.e., each feature has equal weight in the similarity function). This assumption may lead to significant performance degradation if the data set contains many irrelevant features.

In *Nested Generalized Exemplars* (NGE) theory, learning is accomplished by storing objects in Euclidean n-space, E^n, as hyperrectangles (Salzberg, 1991). NGE adds generalization on top of the simple exemplar-based learning. It adopts the position that exemplars, once stored, should be generalized. The learner compares a new example to those it has seen before and finds the most similar, according to a similarity metric which is inversely related to the distance metric (Euclidean distance in n-space). The term *exemplar* (or *hyperrectangle*) is used to denote an example stored in memory. Over time, exemplars may be enlarged by generalization. Once a theory moves from a symbolic space to Euclidean space, it becomes possible to *nest* one generalization inside the other. Its generalizations, which take the form of hyperrectangles in E^n, can be nested to an arbitrary depth, where inner rectangles act as *exceptions* to the outer ones.

EACH (Exemplar-Aided Constructor of Hyperrectangles) is a particular implementation of the NGE technique, where an exemplar is represented by a hyperrectangle. EACH uses numeric slots for feature values of every exemplar. The generalizations in EACH take the form of *hyperrectangles in Euclidean n-space*, where the space is defined by the feature values for each example. Therefore, the generalization process simply replaces the slot values with more general values (i.e., replacing the range $[a, b]$ with another range $[c, d]$, where $c \leq a$ and $b \leq d$). EACH compares the class of a new example with the most similar (shortest distance) exemplar in memory. The distance between an example and an exemplar is computed similarly to the *similarity* function of IBL algorithms, but exemplars and features also have weights in this computation and the result is the distance.

In the *feature partitioning* (FP) 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 Güvenir and Şirin (1996). Learning in CFP is accomplished by storing the objects separately in each feature dimension as disjoint sets of values called *segments*. A segment is expanded through generalization or specialized by dividing it into sub-segments. Classification is based on a weighted voting among the individual predictions of the features, which are simply the class values of the segments corresponding to the values of a test instance for each feature. Description of the same concepts in both FP and OFI are presented in Figure 2. The main difference between FP and OFI techniques is that FP uses disjoint partitioning of feature values into non-overlapping intervals (segments), whereas, OFI allows overlapping of intervals that belong to different classes.

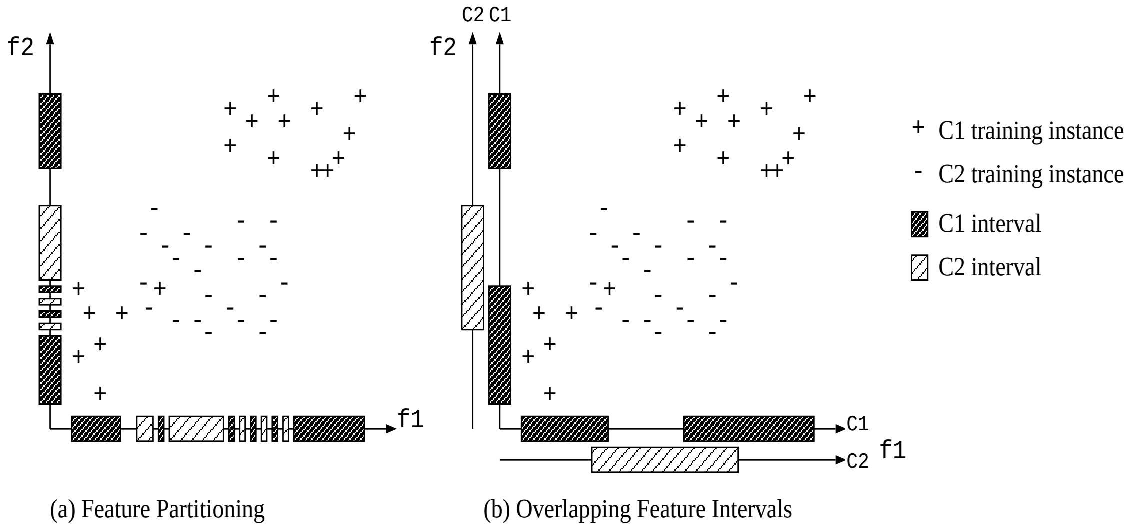

Figure 2: Comparison of representation by Feature Partitioning and Overlapping Feature Intervals for the same dataset.

THE COFI ALGORITHM

The input of the COFI algorithm is a training dataset, and the output is the classification knowledge in the form of overlapping feature intervals. Each instance in the training dataset is represented by a vector. The ith instance e_i is represented as $e_i =< x_{i,1}, x_{i,2}, .., x_{i,n}, c_j >$ where $x_{i,1}, x_{i,2}, .., x_{i,n}$ are the corresponding feature values of features $f_1, f_2, .., f_n$, and c_j is the associated class of the example e_i, where $1 \leq j \leq k$, and k is the total number of the classes.

Learning in COFI

Learning overlapping feature intervals is done by storing the projections of objects separately on each feature-class dimension, and then generalizing these values of the same class into intervals. The basic unit of the representation is an *interval*. An interval is represented by its lower and upper bounds, representativeness count, and its class.

$$I \ = < [lower\ bound,\ upper\ bound],\ class,\ representativeness\ count > .$$

Lower and upper bounds of an 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, that is the instances which have the corresponding feature value between lower and upper bounds, and their class value is the same as the class value of the interval.

In the COFI algorithm, learning is achieved by obtaining the projection of the concepts over each class dimension for each feature. Initially, when the first example is processed, a point interval is constructed. The lower and upper bounds of this interval are equal to the corresponding feature value. It is represented as a point in the feature dimension of the corresponding class.

As an example, suppose that the first training instance of a training dataset e_1 is of class c_1. As shown in Figure 3a, the corresponding feature-class dimensions are updated and a point interval is constructed for c_1 on each feature. In other words, if the first example is $e_1 =< x_{1,1},\ x_{1,2},\ c_1 >$ where $x_{1,1}$ and $x_{1,2}$ 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. The point interval firstly partitions the corresponding class dimension of a 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, c_1. Therefore, the intervals for feature f_1 in Figure 3a are represented as follows:

$< (-\infty,\ x_{1,1}),\ \texttt{UNDETERMINED},\ 0 >$,
$< [x_{1,1},\ x_{1,1}],\ c_1,\ 1 >$, and
$< (x_{1,1},\ \infty),\ \texttt{UNDETERMINED},\ 0 >$.

For the second feature f_2, a similar representation is constructed. If the next training example belongs to another class, 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, then there is a potential for generalization of intervals. An interval may be extended through generalization with other neighboring points or 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. The generalization distance D_f for each feature is determined dynamically as

$$D_f = (current\_max_f - current\_min_f) \times g.$$

Here, $current\_max_f$ and $current\_min_f$ are maximum and minimum values for the feature f seen up to the current training instance, respectively. This range is multiplied by a global parameter g, called *generalization ratio.* The parameter g is selected in the range [0,1]).

The generalization process is applied only to linear features. 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. If the distance between the current training instance and the previously constructed intervals is greater than the D_f, then the training instance constructs a new point interval. If, on the other hand, the training instances falls properly in an interval, then simply the representativeness count of that interval is incremented by one.

Suppose that the f_1 value of the second training instance e_2 is $x_{2,1}$ and the distance between $x_{2,1}$ and the point interval $< [x_{1,1}, x_{1,1}], c_1, 1 >$ is less than the generalization distance D_1. Also suppose that the f_2 value of the second instance e_2 is $x_{2,2}$ and the distance between $x_{2,2}$ and the point interval $< [x_{1,2}, x_{1,2}], c_1, 1 >$ is greater than the generalization distance D_2. That is, for the second example $e_2\ =< x_{2,1},\ x_{2,2},\ c_1 >$,

$$|x_{1,1} - x_{2,1}| \leq D_1 \ \text{ and } \ |x_{1,2} - x_{2,2}| > D_2.$$

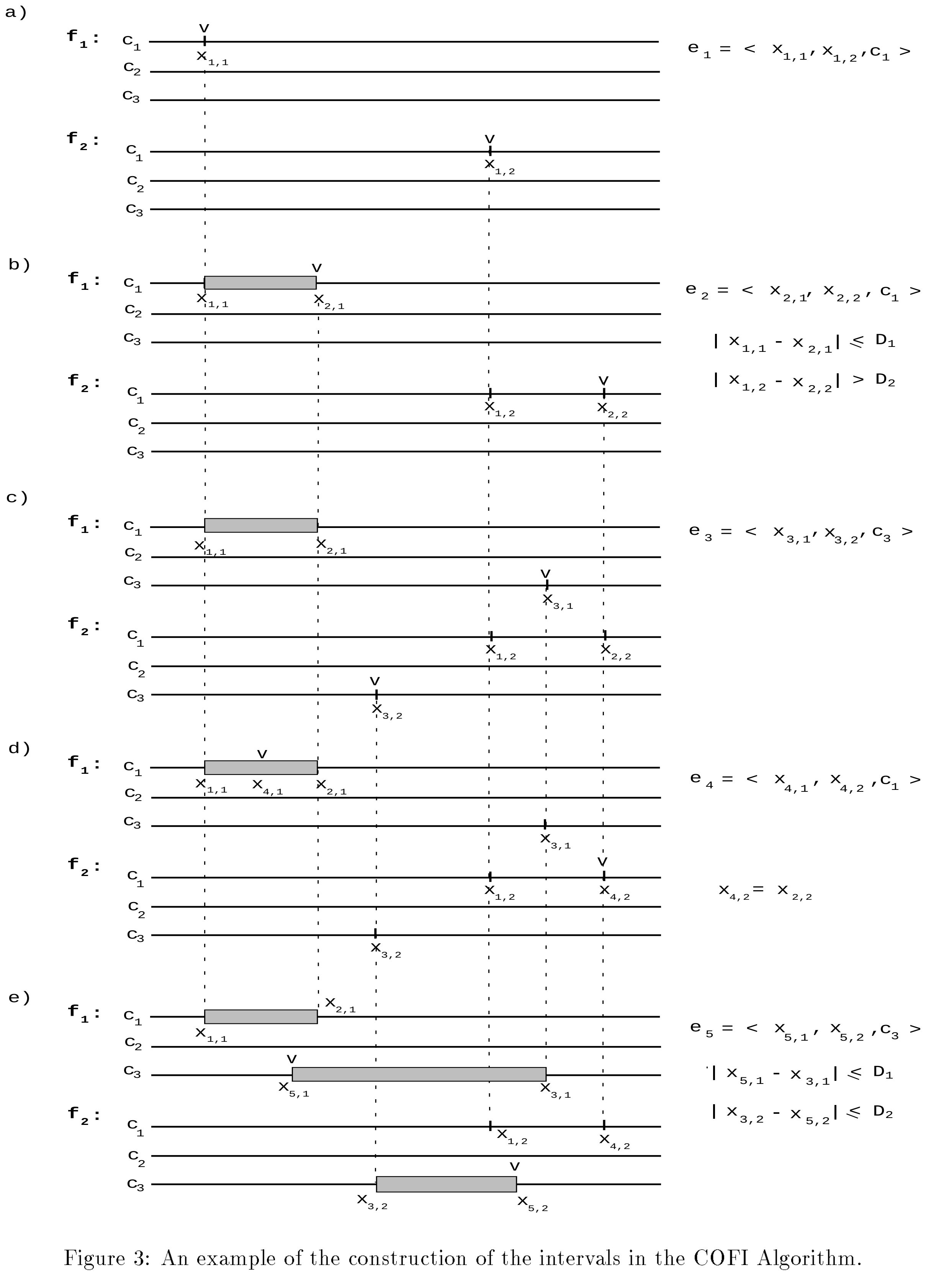

Figure 3: An example of the construction of the intervals in the COFI Algorithm.

For the first feature f_1, generalization will occur on the c_1 dimension of f_1, but no generalization will be done on f_2's class dimension c_1. The generalization process is illustrated in Figure 3b. The COFI algorithm will generalize the interval for $x_{1,1}$ into an extended interval $< [x_{1,1},\ x_{2,1}],\ c_1,\ 2 >$. Here, the representativeness count is also incremented by one, because this interval represents two examples now. Another point interval is constructed in the f_2's corresponding class dimension, so c_1 dimension of this feature will have two point intervals $< [x_{1,2},\ x_{1,2}],\ c_1,\ 1 >$ and $< [x_{2,2},\ x_{2,2}],\ c_1,\ 1 >$, along with three interleaving range intervals of `UNDETERMINED` class.

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 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 3d, then the representativeness count of the interval is incremented by one, that is if the interval is $< [x_{1,1}, x_{2,1}], c_1, 2 >$ and the next instance's f_1, $x_{3,1}$, 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 $< [x_{1,1}, x_{2,1}], c_1, 3 >$. No physical change occurs on the boundaries of the interval but the representativeness count is incremented by one.

Perhaps the most important characteristic 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 behind 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.

Let us show the formation of overlapping intervals through an example. Let the fifth training instance of Figure 3 belongs to c_3. The feature values of this instance is shown in Figure 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 c_3 class dimension of f_1. 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 overlap occurs between the descriptions of class c_1 and c_3. For the feature f_1, overlap occurs between the feature values $x_{5,1}$ and $x_{2,1}$, and for the feature f_2 overlap occurs at the point $x_{1,2}$. In Figure 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 may exist at the same time, especially in medical domains.

Classification in COFI

The classification of a test instance 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 all class dimensions for feature f. The results of the search for feature f is summarized in a vote vector

$$\mathbf{v}_f = < v_{f,1}, v_{f,2}, ..., v_{f,k} > .$$

Here, k is the number of classes and

$$v_{f,c} = \begin{cases} \textit{relative representativeness count} \text{ of } I & \text{if } i_f \text{ falls in interval } I \\ & \qquad \text{of class } c \text{ on feature } f \\ 0 & \text{otherwise.} \end{cases}$$

The final vote vector $\mathbf{v}$ is obtained by summing these vectors:

$$\mathbf{v} = \sum_{f=1}^{n} \mathbf{v}_f = < v_1, v_2, .., v_c, ..v_k > .$$

where $v_c = \sum_{f=1}^{n} v_{f,c}$.

The final classification for the test instance i is the class which received the highest amount of votes:

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

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 value on that feature.

As an example of the classification, suppose that the intervals are formed for all the training set elements as shown in Figure 4. For feature f_1, there exists one interval for c_1 and c_2, and two intervals for class c_3. The corresponding intervals are also shown in Figure 4. For the second feature f_2, there exist two intervals for c_1, one of them is a point interval. There is one interval for each of c_2 and c_3.

In determining the vote of a feature for a class, there are two possibilities. A test instance i may fall into an interval or not. Let the test instance i's first and second feature values $i_{1,1}$ and $i_{1,2}$ be given as shown in Figure 4. To form the vote vectors, firstly relative representativeness count of each matched interval should be computed. As seen in Figure 4, the class counts of c_1, c_2 and c_3 are 100, 80 and 95, respectively.

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

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

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

$$\mathbf{v}_2 = < 0/100, 0/80, 95/95 > = < 0, 0, 1 > .$$

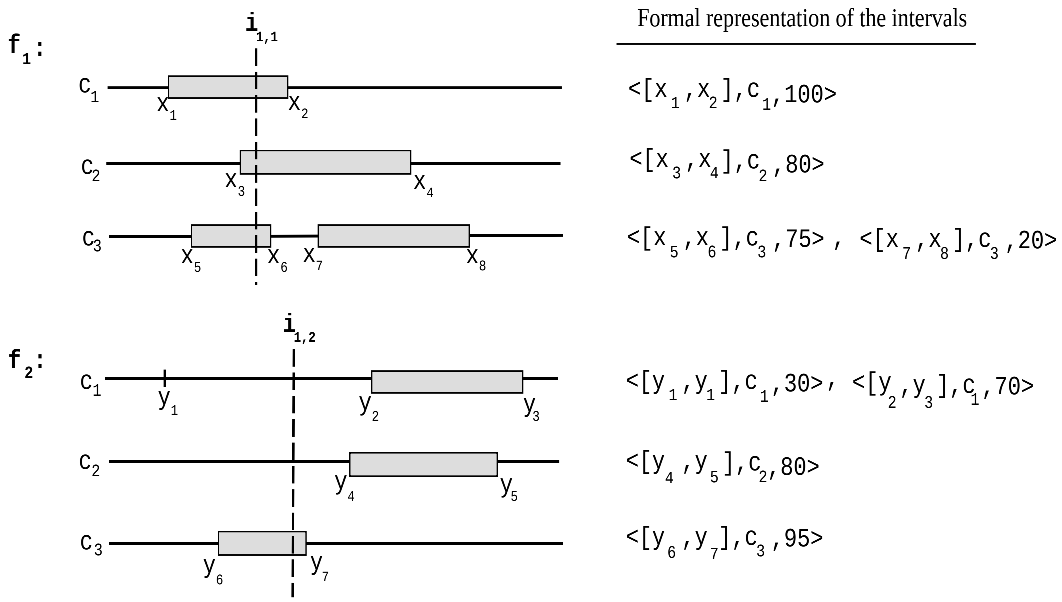

Figure 4: An example of classification in the COFI Algorithm.

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

$$\mathbf{v} \;=\; \mathbf{v}_1 + \mathbf{v}_2 \;= < 1, 1, 0.79 > + < 0, 0, 1 > \,=\, < 1, 1, 1.79 > .$$

Since the vote that class c_3 receives is the highest, the final classification is the class c_3.

EVALUATION OF COFI

In this section, both a complexity analysis and an empirical evaluation of the COFI algorithm is presented. In the complexity analysis, space, training time and testing time complexities of the COFI algorithm are computed. In the empirical evaluation, the COFI algorithm is compared with the k-NN, the NBC and the CFP algorithms on some real-world datasets.

Complexity Analysis of the COFI Algorithm

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

$$O(mn).$$

where m is the total number of training instances and n the total number of features, hence mn is the maximum total number of intervals.

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. For nominal features no generalization is done. In this case, all the intervals are point intervals. In the worst case, 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.

Learning in the COFI algorithm, involves the update of the current concept representation for each training instance. Updating the current representation with a new training instance requires a search for the interval, that the training instance falls in for each feature-class dimension. In our implementation, intervals are stored in an ordered list, and we use a simple linear search. The complexity of this search is proportional to the number of existing intervals currently on the corresponding feature class dimension. In the worst case the number of intervals for one feature is m. Therefore, the time required by the training process for the ith instance is

$$c_t \times n \times k \times (i/k)$$

where c_t is the training constant, k is the number of classes and i/k is the average number of examples for each class up to ith training instance. For all the training instances, this time will be

$$c_t \times m \times n \times k \times (m/k)$$

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

$$O(m^2 n).$$

The classification of a test instance requires a search on each class dimension for all features. Since, in the worst case, there will be m intervals, the complexity of classification of a test instance is equal to the sequential search time on ordered lists which is

$$O(mn).$$

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

Empirical Evaluation of the COFI Algorithm

The COFI algorithm is tested on some real-world datasets which are widely used in the machine learning field. 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 compared the COFI algorithm with the NBC (Naive Bayesian Classifier) (Duda & Hart, 1973), k-NN (k Nearest Neighbor Classifier), and the CFP (Classification by Feature Partitioning) (Guvenir & Sirin, 1996) algorithm. The results of the comparison are given in Table 1.

Our implementation of the NBC algorithm does not make any assumption about the distribution of values for features, such as normal distribution. Therefor, both NBC and k-NN algorithms store all the instances in the memory.

Table 1: Comparison of NBC, k-NN, CFP and COFI in terms of accuracy (%) and memory (number of values stored) on some real-world datasets using leave-one-out evaluation technique.

	NBC		1-NN		CFP		COFI		
Dataset	Acc	Mem	Acc	Mem	Acc	Mem	Acc	Mem	g
Iris	95.33	745	95.33	745	96.00	606	92.67	180	0.020
Glass	57.01	2130	70.09	2130	53.27	5264	57.94	1098	0.014
Horse-colic	80.70	8441	79.61	8441	81.52	2792	79.89	504	0.300
Ionosphere	88.60	12250	86.89	12250	89.46	20552	94.30	280	0.270
Hungarian	83.33	4102	76.53	4102	81.29	1976	85.37	196	0.250
Cleveland	80.53	4228	77.56	4228	84.82	2452	82.83	244	0.100
Wine	93.26	2478	94.94	2478	89.89	5215	96.63	164	0.200

The CFP algorithm is similar to the COFI algorithm. CFP uses feature partitioning instead of overlapping feature intervals. A set of feature values is partitioned into segments of the same class. While COFI assigns weights to intervals, CFP learns a weight for each feature.

In this paper, *leave-one-out cross-validation technique* is used to report the performance of the COFI algorithm. Leave-one-out 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 process is repeated m times and the total number of correct classifications is taken as the leave-one-out cross-validation accuracy of the learning algorithm on the given dataset. The values given for memory sizes in Table 1 are the number of values stored after training with $m-1$ training instances.

It is seen that, the COFI algorithm achieves better results than the other algorithms on four out of seven of these datasets. All the classifiers achieve similar accuracies in Iris, and Horce-colic datasets. The k-NN algorithm outperforms the others on the Glass dataset; while it does poorly on the Cleveland dataset.

The CFP and the COFI algorithm usually achieve similar accuracy results, and they are better than the NBC and k-NN on the average. The COFI algorithm results in better accuracy for Glass, Ionosphere, Hungarian, and Wine datasets than for the CFP 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.

When memory requirements are compared among these algorithms, it is clear that, the COFI algorithm always requires less memory than the other algorithms. The number of intervals stored in memory by the COFI algorithm depends on the generalization ratio and the structure of the dataset, in other words, the distribution of the feature values. Four values are stored in memory for each interval. On the other hand, in the NBC and k-NN algorithms, all the instances are kept in memory verbatim. Therefore, for each instance the feature values and the associated class value should be stored. Table 1 gives 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.

In the CFP algorithm, segments are stored in memory. However, since no overlapping is allowed and sub-partitioning 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 requirements of the CFP algorithm are also shown in Table 1.

CONCLUSION

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.

The COFI algorithm is similar to the NBC and CFP algorithms, in that all of these techniques process each feature separately. All of them use feature based representation, and 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. A version of k-NN algorithm, called k-NNFP (for k Nearest Neighbors on Feature Projections), has also been shown to achieve about the same accuracy as the k-NN algorithm (Akkuş & Guvenir, 1996). The k-NNFP also requires much less memory than that of the k-NN algorithm.

The COFI algorithm is applicable to domains, where each feature can be used in classification independently from others. Holte (1993) has pointed out that the most datasets in the UCI repository are such that, for classification, their features can be considered independently of each other. Also Kononenko (1993) claimed that in the data used by human experts there are no strong dependencies between features because features are properly defined.

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 (Grzymala-Busse, 1987; Quinlan, 1989). Like CFP, the COFI algorithm also ignores the unknown attribute values, which is a natural and plausible approach. Since the value of each attribute is handled separately, this causes no problem.

The behavior of the COFI algorithm to the irrelevant features is also 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 fragmentation of the feature dimensions into large number of small segments. While the COFI algorithm decreases the number of intervals, the CFP algorithm increases its segments of irrelevant features.

The COFI algorithm uses relative representativeness counts for prediction, which 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. Such a normalization is need because datasets usually contain non-homogeneous examples in terms of the number of examples that have the same class value.

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.

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, if only an approximate answer suffices. However, in some techniques, for example in decision trees, the algorithm has to search all features until it reaches to a leaf (Utgoff, 1994). 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 k-NN algorithms, the COFI algorithm uses much less storage, because both the NBC and the k-NN algorithms should keep all feature values separately in the memory to find the probability density function in NBC and to find the distance measure in k-NN for predictions. In the COFI algorithm, intervals should be kept in memory. The memory required for intervals is usually much less than the memory required for the instances themselves.

In summary, 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 learning and classification speed of the algorithm are its important advantages.

REFERENCES

Aha, D.W. 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. *Int. J. Man-Machine Studies*, 36:267-287.

Aha, D.W., Kibler, D., and Albert, M.K. 1991. Instance-based learning algorithms. *Machine Learning*, 6: 37-66.

Akkus, A., and Guvenir, H.A. 1996. K Nearest Neighbor Classification on Feature Projections. *Proc. International Conference on Machine Learning*, ICML'96. Lorenza Saitta (Ed.), pp. 12-19, Morgan Kaufmann, Italy, July.

Duda, R.D., and, Hart, P.E. 1973. *Pattern Classification and Scene Analysis*, New York:

Wiley.

Grzymala-Busse, J.W. 1991. On the Unknown Attribute Values in Learning from Examples. *Proc. Sixth International Symposium Methodologies for Intelligent Systems*, pp. 368-377, October.

Guvenir, H.A., and Sirin, I. 1996. Classification by Feature Partitioning. *Machine Learning*, 23:47-67.

Holte, R.C. 1993. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. *Machine Learning*, 11:63-91.

Kononenko, I. 1993. Inductive and Bayesian Learning in Medical Diagnosis. *Applied Artificial Intelligence*, 7:317-337.

Quinlan, J.R. 1993. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, California.

Quinlan, J.R. 1989. Unknown Attribute Values in Induction, In A. Segre (Ed.), *Proc. 16th International Workshop on Machine Learning*, pp. 164-168, San Mateo, CA:Morgan Kaufmann.

Rendell, L. 1986. A General Framework for Induction and a Study of Selective Induction. *Machine Learning*, 1:177-226.

Salzberg, S. 1991. A nearest hyperrectangle learning method. *Machine Learning*, 6:251-276.

Utgoff, P.E. 1994. An improved algorithm for incremental induction of decision trees. *Machine Learning: Proceedings of the Eleventh International Conference* (pp. 318-325). New Brunswick, NJ: Morgan Kaufmann.

Winston, P.H. 1984. *Artificial Intelligence*, Addison-Wesley, Reading-Massachusetts.