An Algorithm for Classification by Feature Partitioning

İzzet Şirin H. Altay Güvenir

Computer Engineering and Information Science Department
Bilkent University, Ankara 06533 TURKEY
E-mail: {sirin,guvenir}@trbilun.BITNET

Abstract

This paper presents a new methodology for learning from examples, called *Classification by Feature Partitioning* (CFP), which is an inductive, incremental and supervised learning method. Learning in CFP is accomplished by storing the objects separately in each feature dimension as disjoint partitions of values. A partition, which is initially a point in the feature dimension, is expanded through generalization. The CFP algorithm specializes a partition by subdividing it into sub-partitions. It is shown that the CFP algorithm has low sample complexity and training complexity. CFP is also empirically evaluated in three different domains, and the results are compared with *Instance-based learning*, *Nested Generalized Exemplars* and *Decision Tree* techniques.

1 Introduction

Several different representation techniques have been used to describe concepts for supervised learning tasks. Among others, these include decision trees in Quinlan (1986), instance-based representation in Aha and Kibler (1991) and hyperrectangles in Salzberg (1991). The representation of the concepts learned by the exemplar-based learning techniques stores only specific examples that are representatives of other several similar instances. Exemplar-based learning was originally proposed as a model of human learning by Medin and Schaffer [5].

Previous implementations of this approach usually extend the nearest neighbor algorithm, in which some kind of similarity (or distance) metric is used for prediction. Hence, prediction complexity of such algorithms is proportional to the number of instances (or objects) stored.

This paper presents another form of exemplar-based learning, called Feature Partitioning (FP). The FP technique makes several significant improvements over other exemplar-based learning algorithms, where the examples are stored in memory without any change in the representation. For example, IBL algorithms learn a set of instances (a representative subset of all training examples), EACH (Exemplar-Aided Constructor of Hyperrectangles) learns a set of hyperrectangles of the examples. On the other hand, the FP method stores the instances as factored out by their feature values. The program that implements the feature partitioning technique in this paper is called CFP, for Classification by Feature Partitioning. The CFP partitions each feature into segments corresponding to concepts. Therefore, the concept description learned by the CFP is a collection of feature partitions. In other words, the CFP learns a projection of the concept on each feature dimension.

Since the CFP learns projections of the concepts, it does not use any similarity (or distance) metric for prediction. Each feature contributes the prediction process by its local knowledge. Final prediction is based on a voting among the predictions of the features. Since a feature partition can be represented by a sorted list of line segments, the prediction by a feature is simply a search on that sorted list. Therefore, the CFP algorithm significantly reduces the prediction complexity, over other exemplar-based techniques. The strength of the contribution of a feature in the voting process is determined by the weight of that feature. Assigning variable weights to the features modifies the importance of each feature to reflect its relevance for classification. This scheme allows smooth performance degradation when data set contains irrelevant features. Therefore, the CFP algorithm is tolerant to noisy data. The weights (relevancies) of the features are also learned by the CFP algorithm.

The issue of unknown attribute values is an unfortunate fact of real-world data sets, that data often contain missing attribute values. Most of the learning systems, usually overcome this problem by either filling in missing attribute values (with most probable value or a value determined by exploiting interrelationships among the values of different attributes), or looking at the probability distribution of values of attributes. Most common approaches are compared in Quinlan (1993), leading to a general conclusion that some approaches are clearly inferior but no one approach is uniformly superior to others. In contrast, CFP solves this problem very naturally. Since CFP treats each attribute value separately, in the case of an unknown attribute value, it simply leaves the partitioning of that feature intact.

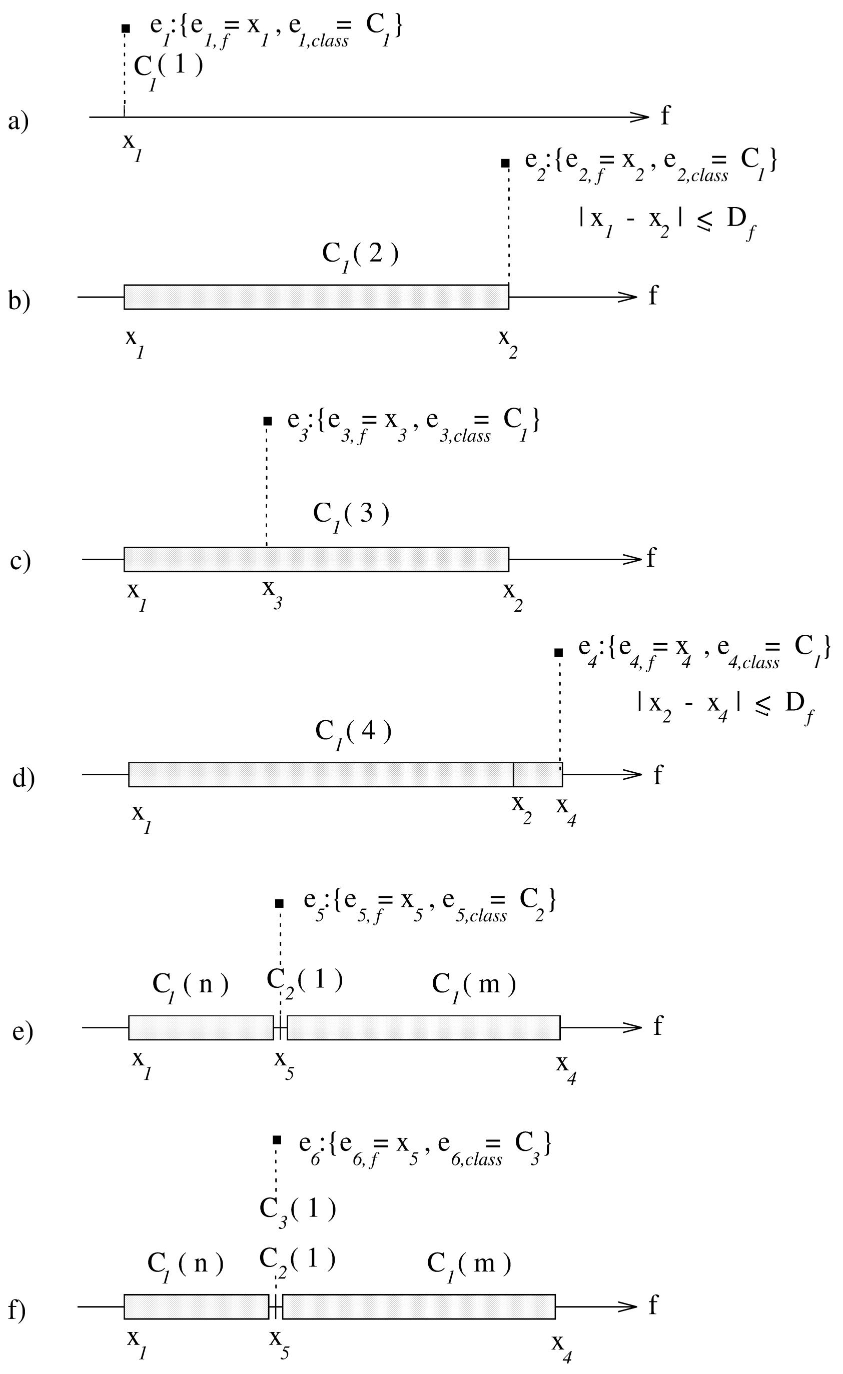
The precise details of the CFP algorithm are described in the next section, and the process of partitioning of a feature dimension is illustrated through an example. Section 3 presents an analysis of the CFP algorithm with respect to PAC-learning theory, and it is shown that the CFP has a low sample and training complexity. In Section 4 the CFP algorithm is compared with other related works. Section 5 presents and empirical evaluation of the CFP algorithm, and a comparison with other techniques on five different data sets. The paper concludes with a a general discussion of the applicability of the CFP and a general evaluation of the algorithm.

2 The CFP Algorithm

The CFP program learns the projection of the concepts over each feature dimension. In other words, the CFP learns partitions of the set of possible values for each feature. An example is defined as a vector of features values plus a label that represents the class of the example. Partition is the basic unit of representation in the CFP algorithm. For each partition, lower and upper bounds of the feature values, the associated class and the number of instances it represents are maintained.

Initially, a partition is a point (lower and upper limits are equal) on the line representing the feature dimension. For instance, suppose that the first example e_1 of class C_1 is given during the training phase (Fig. 1.a). If the value of e_1 for feature f is x_1, that is $e_{1,f} = x_1$ then the set of possible values for feature f will be partitioned into three partitions: $< [-\infty, x_1], undetermined, 0 >$, $< [x_1, x_1], C_1, 1 >$, $< [x_1, \infty], undetermined, 0 >$; where the first element indicates the range of the partition, the second its class, and the third, called the representativeness value, the number of examples represented by the partition.

A partition can be extended through generalization with other neighboring points in the same feature dimension. Assume that the second example e_2 is close to e_1 in feature f and also of the same class. In that case the CFP algorithm will generalize the partition for x_1 into an extended partition: $< [x_1, x_2], C_1, 2 >$, which now represents two examples (see Fig. 1.b). Generalization of a range partition is



e_1:{$e_{1,f} = x_1$, $e_{1,class} = C_1$}
$C_1(1)$
a)
x_1
f
e_2:{$e_{2,f} = x_2$, $e_{2,class} = C_1$}
$|x_1 - x_2| \leqslant D_f$
$C_1(2)$
b)
x_1
x_2
f
e_3:{$e_{3,f} = x_3$, $e_{3,class} = C_1$}
$C_1(3)$
c)
x_1
x_3
x_2
f
e_4:{$e_{4,f} = x_4$, $e_{4,class} = C_1$}
$|x_2 - x_4| \leqslant D_f$
$C_1(4)$
d)
x_1
x_2
x_4
f
e_5:{$e_{5,f} = x_5$, $e_{5,class} = C_2$}
$C_1(n)$
$C_2(1)$
$C_1(m)$
e)
x_1
x_5
x_4
f
e_6:{$e_{6,f} = x_5$, $e_{6,class} = C_3$}
$C_3(1)$
$C_1(n)$
$C_2(1)$
$C_1(m)$
f)
x_1
x_5
x_4
f

Figure 1: Partitioning of a feature dimension.

illustrated in Fig. 1.d.

Since partitions are disjoint the CFP algorithm pays attention to avoid over generalization. In order to generalize a partition in feature f to cover a point, the distance between them must be less than a given generalization limit (D_f). Otherwise, the new example is stored as another point partition in the feature dimension f.

If the feature value of a training example falls in a partition with the same class, then simply the representativeness value (number representing the examples in the partition) is incremented by one (see Fig. 1.c).

If the new training example falls in a partition with a different class than that of the example, the CFP algorithm specializes the existing partition by dividing it into two range partitions and inserting a point partition (corresponding to the new example) in between them (see Fig. 1.e, f). When a partition is divided into two partitions, it is important to distribute the representativeness value of the old partition to the newly formed partitions. The CFP distributes the representativeness of the old partition among the new ones in proportion to their sizes. For instance, the representativeness value of the newly formed partitions in Fig. 1.e will be

$$n = 4\frac{x_5 - x_1}{x_4 - x_1},$$

$$m = 4\frac{x_4 - x_5}{x_4 - x_1}.$$

In terms of production rules, the partitioning in Figure 1.f can be represented as:

if $e_f \geq x_1$ and $e_f < x_5$
then $e_{class} = C_1$

if $e_f = x_5$
then $e_{class} = C_1$

if $e_f > x_5$ and $e_f \leq x_4$
then $e_{class} = C_1$

The CFP algorithm pays attention to the disjointness of the partitions. However, partitions may have common boundaries in this case, the CFP algorithm uses the representativeness values of the partitions to determine class value. For example, in Fig. 1.f at $e_f = x_5$, three classes C_1, C_2 and C_3 are possible, but since the total representativeness of the class C_1 is 4 and the that of the other classes is 1, the prediction for the feature f is C_1.

```
train(Training Set):
begin
    foreach $e$ in Training Set
        foreach feature $f$
            if class of partition($f, e_f$) $= e_{class}$
                then $w_f = (1 + \Delta)w_f$
                else $w_f = (1 - \Delta)w_f$
        foreach feature $f$
            update-feature-partitioning($f, e_f$)
end
```

Figure 2: Training algorithm of the CFP.

```
prediction($e$):
begin
    foreach feature $f$
        $c$ = class of partition($f, e_f$)
        $vote_c = vote_c + w_f$
    return class $c$ with highest $vote_c$.
end
```

Figure 3: Prediction process of the CFP.

The training process in CFP algorithm has two steps: learning of feature weights and feature partitions (Fig. 2). 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 incremented by Δ (global feature weight adjustment rate) percent; otherwise, it is decremented by the same amount.

The prediction in the CFP is based on a voting taken among the predictions made by each feature separately (Fig. 3). For a given instance e, the prediction based on a feature f is determined by the value of e_f. If e_f falls properly within a partition with a known class then the prediction is the class of that partition. If e_f falls in a point partition then among all the partitions at this point the one with the highest representativeness value is chosen. If e_f falls in a partition with no known class value, then no prediction for that feature is made. The effect of the prediction of a feature in the voting is proportional with the weight of that feature. All feature weights are initialized to one before the training process begins. The predicted class of a given instance is the one which receives the highest amount of votes among all feature predictions.

```
update-feature-partitioning($f, e_f$):
begin
    if class of partition($f, e_f$) = $e_{class}$
      increment representativeness value of partition($f, e_f$)
    else {different class}
      if partition($f, e_f$) is a point partition
        insert-new-partition($f, e_f$)
      else {partition($f, e_f$) is a range partition}
          if class of partition($f, e_f$) is not *undetermined*
            subdivide-partition(partition($f, e_f$), $e_f$)
          else {try to generalize}
              if the nearest *partition* to left or right in $D_f$ distance has the class $e_{class}$
                generalize(*partition*,$e_f$)
              else {there are no partitions in $D_f$ distance with the same class as $e$}
                  insert-new-partition($f, e_f$)
end
```

Figure 4: Updating a feature partition.

The second step in the training process is to update the partitioning of each feature using the given training example (Fig. 4). If the feature value of a training example falls in a partition with the same class, then simply its representativeness value is incremented. If the new feature value falls in a partition with a different class than that of the example and this partition is a point partition, then a new point partition (corresponding to the new feature value) is inserted next to the old one. Otherwise, if the class of the partition is not undetermined, then the CFP algorithm specializes the existing partition by dividing it into two range partitions and inserting a point partition (corresponding to the new feature value) in between them. On the other hand, if the example falls in an undetermined partition, the CFP algorithm tries to generalize a near partition with the feature value. If one of the nearest partitions to the left and the right of the new example is in D_f distance and of the same class as the example, then it is generalized to cover the new feature value. Otherwise, a new point partition that corresponds to the new feature value, is inserted.

In order to see the form of the resulting concept descriptions learned by the CFP algorithm, let us consider a domain with two features, f_1 and f_2. Assume that during the training phase, positive (+) instances with f_1 values in $[x_{11}, x_{12}]$ and f_2 values in $[x_{23}, x_{24}]$, and negative (-) instances with f_1 values in $[x_{13}, x_{14}]$ and f_2 values in $[x_{21}, x_{22}]$ are given. The resulting concept description is shown in Fig. 5.

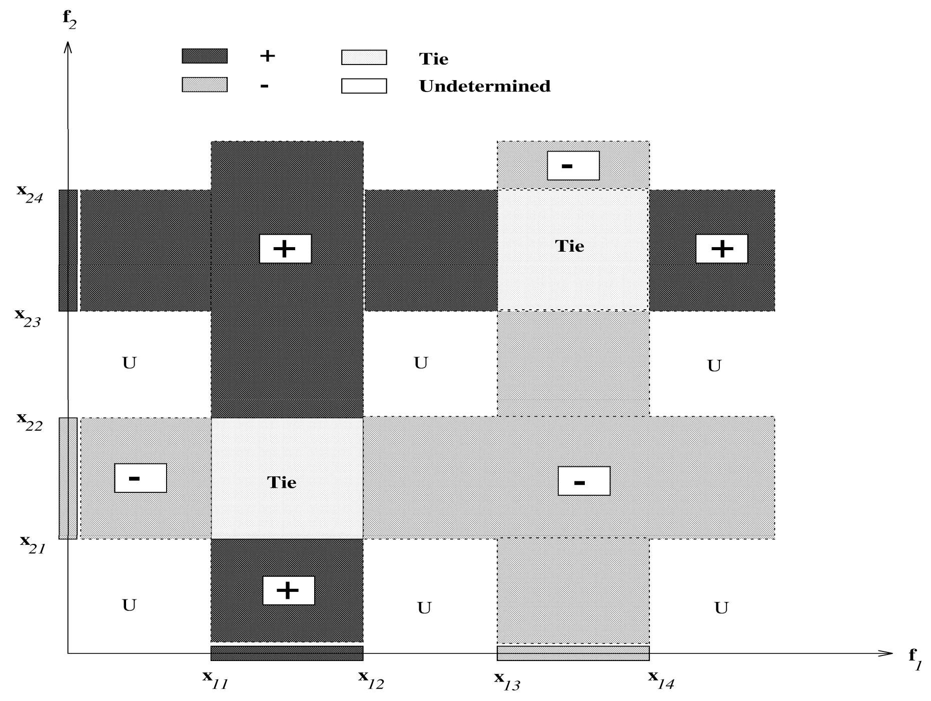


Figure 5: An example concept description in a domain with two features.

For test instances which fall into the region $[-\infty, x_{11}][x_{23}, x_{24}]$, for example, feature f_1 has no prediction, while feature f_2 predicts as class $(+)$. Therefore, any instance falling in this region will be classified as $(+)$. On the other hand, for instances falling into the region $[-\infty, x_{11}][-\infty, x_{21}]$, for example, the CFP algorithm does not commit itself to any prediction.

The description of the class $+$ shown in Fig. 5 can be written in 3-DNF as:

class $+$:

$(x_{11} \leq f_1 \ \& \ f_1 \leq x_{12} \ \& \ f_2 < x_{21})$ or
$(x_{11} \leq f_1 \ \& \ f_1 \leq x_{12} \ \& \ f_2 > x_{22})$ or
$(x_{23} \leq f_2 \ \& \ f_2 \leq x_{24} \ \& \ f_1 < x_{13})$ or
$(x_{23} \leq f_2 \ \& \ f_2 \leq x_{24} \ \& \ f_1 > x_{14})$

Or, more compactly;

class $+$:

$[(x_{11} \leq f_1 \leq x_{12}) \ \& \ (f_2 < x_{21}$ or $x_{22} < f_2)]$ or
$[(x_{23} \leq f_2 \leq x_{24}) \ \& \ (f_1 < x_{13}$ or $x_{14} < f_1)]$

Similarly, the description for the negative examples can be written as:

class $-$:

$[(x_{13} \leq f_1 \leq x_{14})$ & $(f_2 < x_{23}$ or $x_{24} < f_2)]$ or
$[(x_{21} \leq f_2 \leq x_{22})$ & $(f_1 < x_{11}$ or $x_{12} < f_1)]$

The CFP does not assign any classification to an instance, if it could not determine the appropriate class value for that instance. This may result from having seen no instances for a given set of values or having a tie between two or more possible contradicting classifications. In case of different weight values for the features, the ties are broken in favor of the class predicted by the features with the highest weights during the voting process (with equal feature weights, it corresponds to the majority voting scheme).

3 Theoretical Evaluation of the CFP

This section presents an analysis of the CFP algorithm with respect to *PAC-learning* theory [9]. The intent of the PAC (Probably Approximately Correct) model is that successful learning of an unknown target concept should entail obtaining, with high probability, that it is a good approximation of the concept. The two criticisms most often leveled at the PAC model by AI researchers interested in empirical machine learning are (1) the worst-case emphasis in the model makes it unusable in practice and (2) the notions of target concepts and noise-free training data are too restrictive in practice [3].

The analysis of the CFP shows that, it is applicable to a large class of concepts, and requires small number of examples and a small amount of memory to learn a given concept, compared to many other similar algorithms. Another outcome of the analysis is that, the CFP has a lower learning complexity than other similar techniques.

Since the classification in the CFP is based on a voting taken among the individual classifications of each attribute, it can learn a concept if each attribute, independently from other attributes, can be used in the classification. We will define what we mean by "learn" in a way that preserves the spirit of the Valiant (1984) definition of learn-ability, but modifies it for the voting based classification used in the CFP. To do this we first determine the minimum number of training instances to learn a given concept. Using this sample complexity we derive the training complexity of the CFP algorithm. In the following analysis we assume that all feature values are normalized to the interval [0,1].

Definition. Let X be a subset of $\Re^n$ with a fixed probability distribution and d is positive integer less than or equal to n. A subset S of X is an $< \varepsilon, \gamma, d > -net$

for X if, for all x in X, with probability greater than γ, there exist an s in S such that $|s_j - x_j| < \varepsilon$ at least for d values of j ($1 \leq j \leq n$).

Lemma 1. Let ε, δ, and γ be fixed positive numbers less than one and d is positive integer less than or equal to n. A random sample S containing $m > (\lceil 1/\varepsilon \rceil/\gamma) \times (n \ln 2 + \ln(\lceil 1/\varepsilon \rceil/\delta))$ instances, drawn according to any fixed probability distribution from $[0,1]^n$, will form an $< \varepsilon, \gamma, d >$-net with confidence greater than $1 - \delta$.

Proof. We prove this lemma by partitioning the unit interval for each feature dimension, into k equal length sub-intervals, each with length less than ε, such that all pairs of points[1] in the sub-interval are within ε distance of each other. The idea of the proof is to guarantee that, with high confidence, at least for d dimensions out of n, each of k sub-intervals contains at least one point of m instances, with sufficient probability.

Let $k = \lceil 1/\varepsilon \rceil$, S_{1f} be the set of sub-intervals with probability greater or equal to γ/k and S_{2f} be the set of remaining sub-intervals of a dimension f. The probability that an arbitrary point in $[0,1]$ will not lie in a selected sub-interval of S_{1f} is $(1-\gamma/k)$. The probability that none of the m sample points will lie in a selected sub-interval of S_{1f} is $(1-\gamma/k)^m$. Therefore, the probability that any sub-interval of S_{1f} is excluded by all m instances is at most $p = k(1 - \frac{\gamma}{k})^m$.

The probability that, for more than $n - d$ dimensions, any sub-interval of S_1's are excluded by all m instances is at most $\sum_{i=n-d+1}^{n} C(n,i)p^i$.[2] To make sure this probability is small, we force it to be less than δ, that is,

$$\sum_{i=n-d+1}^{n} C(n,i)p^i < \delta.$$

Recall the *binomial theorem*: $(a+b)^n = \sum_{i=0}^{n} C(n,i)a^i b^{n-i}$. With $a = p$ and $b = 1$, $\sum_{i=0}^{n} C(n,i)p^i = (p+1)^n$. Since n is a positive integer, $(p+1)^n - 1 = \sum_{i=1}^{n} C(n,i)p^i$ and it is greater than $\sum_{i=n-d+1}^{n} C(n,i)p^i$, our requirement can be written as

$$(p+1)^n - 1 < \delta.$$

On the other hand, $(1-\gamma/k)^m < e^{-m\gamma/k}$ and, since the value of p is greater than zero and less than one, $2^n p > (p+1)^n - 1$. If we solve the requirement $2^n k e^{-m\gamma/k} < \delta$, for m, and substitute $\lceil 1/\varepsilon \rceil$ for k, it yields

$$m > \lceil 1/\varepsilon \rceil/\gamma \times (n \ln 2 + \ln(\lceil 1/\varepsilon \rceil/\delta)).$$

[1] a point here represents the value of an instance for a feature for that dimension

[2] $C(n,r)$ represents the number of combinations of n things taken r at a time.

Consequently, with confidence greater than $1 - \delta$, each sub-interval in S_{1f} of d or more dimensions, contains some sample point of an instance of S. $\Box$

Theorem 1. Given ε, δ, and γ fixed positive numbers less than one and a sample set S with n features. If for $\lceil\frac{n+1}{2}\rceil$ of features of the elements of S form an $<\varepsilon, \gamma, \lceil\frac{n+1}{2}\rceil>$-net then, the CFP algorithm with equal feature weights and generalization limit $D_f \geq 2\varepsilon$ for all features, will learn a concept C for S with confidence $1 - \delta$.

Proof. Since, the CFP algorithm does not use distance metric for classification, the idea of the proof is to ensure that the CFP can construct ε length partitions with high confidence, at least one of the m sample instances lies in each sub-intervals of $\lceil\frac{n+1}{2}\rceil$ features with sufficient probability. The CFP algorithm employs a majority voting scheme in the classification. Hence, only $d = \lceil\frac{n+1}{2}\rceil$ of the features must agree on the classification. If we follow the proof of the *lemma 1*, if S form an $<\varepsilon, \gamma, d>$-net, then it guarantees that each sub-interval contains at least one instance of S with high confidence. The CFP algorithm will generalize two points into one partition, if the distance between them is less than or equal to D_f. Therefore, if $D_f \geq 2\varepsilon$ then the points will be generalized into one partition, corresponding to a projection of the concept on that feature. $\Box$

Theorem 2. Given ε, δ, and γ fixed positive numbers less than one. If random sample S with n features forms an $<\varepsilon, \gamma, \lceil\frac{n+1}{2}\rceil>$-net with confidence greater than $1 - \delta$, then CFP with $D \geq 2\varepsilon$ constructs at most $n\lceil 1/\varepsilon\rceil$ partitions.

Proof. Since S is an $<\varepsilon, \gamma, \lceil\frac{n+1}{2}\rceil>$-net with with confidence greater than $1-\delta$, each feature line is divided in to ε length sub-intervals and each one contains at least one sample point and the CFP algorithm constructs at most one (due to $D \geq 2\varepsilon$) partition for each sub-interval. Thus, for n features, the CFP constructs at most $n\lceil 1/\varepsilon\rceil$ partitions. $\Box$

Theorem 3. Given ε, δ, and γ fixed positive numbers less than one. If random sample S is an $<\varepsilon, \gamma, \lceil\frac{n+1}{2}\rceil>$-net with confidence greater than $1 - \delta$, then classification complexity of the CFP with $D \geq 2\varepsilon$ is $O(n \log(\lceil 1/\varepsilon\rceil))$ and the training complexity is for m sample instances is $O(mn \log(\lceil 1/\varepsilon\rceil))$.

Proof. Proof of the theorem 2 shows, that the CFP constructs at most $\lceil 1/\varepsilon\rceil$ partitions for each feature. In CFP algorithm the classification is composed of a search and a voting. The complexity of the search operation is $O(\log(\lceil 1/\varepsilon\rceil))$ for each feature. Since the complexity of voting is $O(n)$, the classification complexity of the CFP algorithm is $O(n \log(\lceil 1/\varepsilon\rceil))$ for n features. Consequently, with m training instances, the training complexity of the CFP algorithm is $O(mn \log(\lceil 1/\varepsilon\rceil))$. $\Box$

The classification process in exemplar-based learning algorithms which use some form of the nearest neighbor algorithm (such as EACH and IBL) involves computing the Euclidean distance (or similarity) of the instance to each stored exemplar in each dimension. Therefore, if there are M exemplars stored in the memory, and n features are used, then the complexity of the classification is $O(nM)$. On the other hand, since the partitions are naturally sorted for each feature dimension, the classification process in the CFP algorithm is only $O(n \log M)$. This feature of the CFP algorithm significantly reduces the classification complexity.

Another important feature of the CFP algorithm is its low memory requirement. Since, the CFP learns each feature partitions independently, number of the partitions for each feature may be different. If appropriate D_f generalization limits are chosen the CFP may significantly reduce memory requirement. The selection of D_f values is domain dependent, and is an optimization problem and can be treated as an optimization problem.

4 Related Work

This section briefly reviews some of the related approaches, and compares them with the CFP algorithm.

Instance-Based Learning: The primary output of IBL algorithms is a *concept description*, which is a function that maps instances to concepts. Instance-based learning technique [1], has been 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. Both IB1 and IB2 do not remove any instance from concept description after it had been stored. IB3 employs a significance test (i.e. acceptable or significantly poor) to determine which instances are good classifiers and which ones are believed to be noisy. IB3 accepts an instance if its classification accuracy is significantly greater than the observed frequency of its class, and removes the instance from concept description if its accuracy is significantly less. Confidence intervals are used to determine whether an instance is acceptable, mediocre, or noisy.

An instance-based concept description includes a set of stored instances and some information concerning their past performance during the training process (e.g. number of correct and incorrect classification predictions). The final set of instances can change after each training process. However, IBL algorithms do not construct intensional concept descriptions (or do not make generalization). Instead, concept descriptions are determined by how IBL algorithm's *similarity* and *classification* functions use the current set of saved instances. 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 a long with the set of instances.

IBL algorithms assume that similar instances 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 (e.g. by having equal weight in similarity function). This assumption may lead significant performance degradation if the data set contains many irrelevant features.

Nested Generalized Exemplars: In nested generalized exemplars (NGE) theory learning is accomplished by storing objects in Euclidean n-space, E^n, as hyperrectangles [8]. NGE is also a variation of exemplar-based learning. In the simplest form of the exemplar-based learning, every example is stored in memory, with no change in representation (or without generalization), as in IB1 algorithm presented above. 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 modified (due to generalization) from their original forms. This similar to the generalizations of partitions in the CFP algorithm.

Once a theory moves from a symbolic space to a Euclidean space, it becomes possible to *nest* generalization one 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. The CFP algorithm avoids over-generalizations, therefore the partitions are not nested, and there are no exceptions.

EACH (Exemplar-Aided Constructor of Hyperrectangles) is a particular implementation of the NGE technique [8]. EACH uses numeric slots for feature values of 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, generalization process simply replaces the slot values with more general values (i.e., replacing range of values [a, b] with another range [c, d], where $c \leq a$ and $d \geq b$).

Decision Tree: The decision tree is a well-known representation for classification tasks. This representation has been used in a variety of systems, among them the most famous are ID3 [6] and its extension C4.5 [7] of Quinlan.

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.

A decision tree is global for each attributes, in other words each non-leaf decision node may specify some test on any one of the attributes. In some sense there is some similarity between decision tree representation and feature partitions. In one view, the CFP algorithm can be seen to produce decision trees (each tree can only specifies some test on only one attribute) for each attribute and final prediction of a case is combination of the predictions of the each local trees. As far as the prediction complexity is concerned, decision tree prediction has a low complexity as the CFP algorithm.

An important difference between decision tree approach and other approaches mentioned above, including CFP, is that the classification performance of these systems does not depend critically on any small part of the model. In contrast, decision trees are much more susceptible to small alterations.

5 Empirical Evaluation of the CFP

The CFP algorithm has been tested using real and artificial data from five different problem domains: (1) *classifying iris flowers*, (2) *predicting a recurrence of cancer*, (3) *Hungarian database* from Hungarian Institute of Cardiology, (4) *Cleveland database* from Cleveland Clinic Foundation, and (5) *Classifying waveforms.* The use of real data in these tests provide a measure of the system's accuracy on noisy and incomplete data sets, and most importantly, allowed comparisons between CFP and other systems. Below, each data set is described briefly and experimental results are presented.

Iris Flowers: Iris flowers data set from Fisher [2] consists of four integer-valued features and a particular species of iris. There are three different species (classes): *virginica*, *setosa*, and *versicolor*. The data set contains 150 instances. The accuracy of the CFP in Table 1 was obtained for $\Delta = 0.015$, and D_i limits were 0.6, 0.1, 0.4, 0.1, respectively.

Breast Cancer: Breast Cancer data set contains 273 patient records. All the patients underwent surgery to remove tumors, all of them were followed up five years later. The objective here is to predict whether or not breast cancer would recur during that five year period. The data set contains nine variables that were measured, including both numeric and binary values. The prediction is binary : either the patient did suffer a recurrence of cancer or not. The accuracy of the CFP in Table 1 was obtained for $\Delta = 0.07$, and D_i limits were 4, 0, 6, 0.5, 0.1, 0.5, 0.5, 0.5, 4, respectively.

Medical Diagnosis: The CFP was tested on widely used two medical databases, namely the Cleveland and Hungarian databases. The Cleveland and Hungarian data sets contain heart disease diagnoses collected from the Cleveland Clinic Foundation

Table 1: Comparison of CFP with EACH and C4.5 in terms of accuracy (%)

Database	EACH	C4.5	CFP
Iris	95.3	95.3	96.7
Breast Cancer	77.6	70.1	77.5

Table 2: Comparison of CFP with IBL and C4 algorithms in terms of accuracy (%)

Database	Training Set Size	Test Set Size	IB1	IB2	IB3	C4	CFP
Hungarian	250	44	58.7	55.9	80.5	78.2	82.3
Cleveland	250	53	75.7	71.4	78.0	75.5	84.0
Waveform	300	500	75.2	69.6	73.8	70.7	76.0

and Hungarian Institute of Cardiology, respectively. A diagnosis is described by 13 numeric-valued features (e.g. age, fasting blood sugar level etc.). The objective here is to determine whether a patient has a heart disease. The Cleveland data set consists of 303 instances and the Hungarian data set consists of 294 instances. The accuracy of the CFP for the Hungarian database in Table 2 was obtained for $\Delta = 0.02$, and D_i limits were 1.3, 0, 0.2, 19.1, 36.1, 0, 0.6, 22.9, 0, 0.6, 0.03, 0.1, 0.9, respectively. The accuracy for the Cleveland database in Table 2 was obtained for $\Delta = 0.025$, and D_i limits were 7, 0.3, 0.4, 9, 11, 0.3, 0.4, 7, 0.3, 0.4, 0.4, 0.4, 0.4, respectively.

Classifying Waveforms: The waveform data set is artificial and consists of 21 numeric-valued features, which have values between 0 and 6. there are three different types of waveforms and they are equally distributed. The objective here is to determine the type of waveform. Each feature includes noise (with mean 0 and variance 1). Out of 800 instances in the data set, 300 were used in the training. The CFP achieved 76.0% accuracy on waveform database with $\Delta = 0.02$. Generalization distance limits for the waveform database were 0.2, 0.3, 0.6, 0.5, 0.8, 0.9, 1.0, 0.9, 0.6, 0.6, 0.7, 0.6, 0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.5, 0.5, 0.4, respectively.

The reported results of the EACH [8] and measured results of the C4.5 and CFP algorithms are shown in Table 1. To allow for proper comparisons, the experimental design used was the same as that used by Salzberg (1991). For Breast Cancer data set, for each trial, the examples were divided into a training set and test set. 70 % of the examples were randomly chosen for training set. Four different trials were run, and the final results are an average of those trials. For Iris data set the leaving-one-out cross-validation accuracy estimation technique has been used. A robust estimate of accuracy on unseen examples can be obtained by this technique. Cross-validation involves removing mutually exclusive test sets of examples from the data set. 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 leaving-one-out method involves removing exactly one example from the data and training on the remaining examples. The technique is repeated for every example in the data set and accuracy is measured across all examples.

The performance of the CFP algorithm is also compared with the reported accuracy of the instance-based learning algorithms [1]. All results reported in Table 2 were averaged over 50 trials. The training and test sets were always disjoint. The instances were drawn randomly from the data sets.

In these experiments we noticed that the performance was not sensitive to the small changes in the D_f settings. For binary-valued attributes distance parameter was set to zero for no generalization. The feature weight adjustment rate and the generalization limits are domain dependent. In these experiments their values are determined by trial and error, separately for each application domain.

6 Conclusion

We have presented a new methodology of learning based on feature partitioning, called CFP. It is an inductive, incremental and supervised learning method. The CFP learns a partitioning of values for each feature of the application domain. The CFP algorithm is applicable to concepts, where each feature, independent of other features, can classify the concept. For domains, where most of the attributes are discrete, the performance of the CFP depends on observed frequency of the concepts.

This approach is a variant of algorithms that learn by projecting into one feature dimension at a time. For example, ID3 learns in that greedy manner while building a conjunction. The novelty of CFP is that it retains a feature-by-feature representation and uses voting to categorize. Algorithms that learn by projecting into one dimension at a time are limited in their ability to find complex concepts.

The CFP makes significant modifications to the exemplar-based learning algorithms. The analysis of the CFP shows that, it is applicable to a large class of concepts, and requires small number of examples and a small amount of memory to learn a given concept, compared to many other similar algorithms. Another outcome of the analysis is that, the CFP has also a low training complexity.

Another important improvement is the natural handling of unknown attribute values. Most of the systems use *ad hoc* methods for handling unknown attribute values. Since the value of each attribute is handled separately, attributes with unknown values are simply ignored by the CFP.

The CFP will clearly fail in some cases. For example, if the projection of concepts on an axis are overlapping each other, the CFP constructs many partitions of different classes next to each other. In that case, the accuracy of classification depends on the observed frequency of the concepts in the database.

The CFP uses feature weights to cope with irrelevant attributes. Introducing feature weights protects the algorithm's performance, when application domain has irrelevant or not equally relevant attributes. The concept of feature weights have also been used successfully in other similar systems [4, 8]. In the CFP the feature weights are dynamically adjusted according to the global Δ adjustment rate, which is an important parameter for the predictive accuracy of the algorithm. Another important component of the CFP is generalization limit for each attribute, which controls the generalization process. The Δ adjustment rate and the generalization limits are domain dependent parameters to the CFP, and their selection affects the performance of the algorithm. Determining the best values for these parameters is an optimization problem for a given domain.

References

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

[2] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

[3] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.

[4] J. D. Kelly and L. Davis. A hybrid genetic algorithm for classification. In *The proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 645–650, 1991.

[5] D. L. Medin and M. M. Schaffer. Context theory of classification learning. *Psychological Review*, 85:207–238, 1978.

[6] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

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

[8] S. Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, 6:251–276, 1991.

[9] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.