EMPIRICAL EVALUATION OF THE CFP ALGORITHM

İZZET ŞİRİN and **H. ALTAY GÜVENİR**
Computer Engineering and Information Science Department
Bilkent University, Ankara 06533 TURKEY
E-mail: {sirin, guvenir}@firat.bcc.bilkent.edu.tr

ABSTRACT

This paper presents a new methodology for concept 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 is expanded through generalization or specialized by dividing it into subpartitions. An empirical evaluation of the CFP on real-world data sets is presented.

1. Introduction

This paper presents a form of exemplar-based learning, called *Classification by Feature Partitioning* (CFP). In exemplar-based learning examples are stored in memory verbatim. The CFP technique makes several significant improvements over other exemplar-based learning algorithms. For example, IBL[1] algorithms learn a set of instances which is a representative subset of all training examples. Another algorithm called EACH[5] learns a set of hyperrectangles of the examples. On the other hand, the CFP method stores the instances as factored out by their feature values. The CFP partitions each feature into segments corresponding to concepts. Therefore, the concept description learned by the CFP is a collection of feature partitions.

Each feature contributes the classification process by its local knowledge. Final classification is based on a voting among the predictions of the features. The CFP algorithm significantly reduces the classification 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.

Most real-world data sets contain missing attribute values. Previous learning systems usually overcome this problem by either filling in missing attribute values, or looking at the probability distribution of values of attributes. Most common approaches are compared in Quinlan (1993), leading to a general conclusion that 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.

In the next section the CFP algorithm is described (precise details are given Şirin (1993)). The voting process of the CFP is illustrated through an example. Section 3 presents an empirical evaluation of the CFP algorithm on various real-world data sets. The final section discusses the applicability of the CFP algorithm.

```
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 + Δ)w_f
              else w_f = (1 − Δ)w_f
            update-feature-partitioning(f, e_f)
    end
```

Figure 1: Training algorithm of the CFP

2. The CFP Algorithm

An example is defined as a vector of feature values plus a label that represents its class. The CFP algorithm learns partitions of the set of possible values for each feature. For each partition, lower and upper bounds of the feature values, its 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. Suppose that the first example e_1 of class C_1 is given during the training phase. If the value of e_1 for feature f is x_1, then the set of possible values for feature f will be partitioned into three partitions: $< [-\infty, x_1], U, 0 >$, $< [x_1, x_1], C_1, 1 >$ and $< [x_1, \infty], U, 0 >$; where the elements indicate the range of the partition, its class*, and the representativeness value†, respectively. A partition can be extended through generalization with other neighboring points in the same feature dimension. If the second example e_2 is close to e_1 in feature f and also of the same class, the CFP algorithm will generalize the partition for x_1 into a range partition: $< [x_1, x_2], C_1, 2 >$, which represents two examples.

The training process of the CFP algorithm has two steps: learning feature weights and learning feature partitions (Fig. 1). 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, the weight of that feature is incremented by Δ (global feature weight adjustment rate) percent; otherwise, it is decremented by the same amount.

The second step in the training process is to update the partitioning of each feature-space using the given training example. 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 range partition with a different class than that of the example, the CFP algorithm specializes the existing partition by dividing it into two subpartitions 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

*U represents undetermined class

†representativeness value indicates the number of examples represented by a partition.

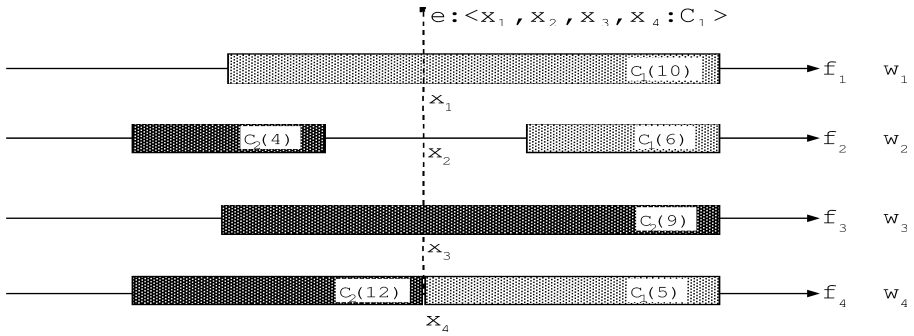

Figure 2: Classification process of the CFP

point partition that corresponds to the new feature value, is inserted.

The classification process of the CFP is based on a voting taken among the predictions made by each feature separately. 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 determined class then the prediction is the class of that partition. If e_f falls into the border of more than one partitions, 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 predictions. Figure 2 shows an example of classification process of the CFP. Consider a test example e of class C_1 with feature values x_1, x_2, x_3, and x_4. The prediction of the first feature is C_1. The second feature does not predict any class value (*undetermined*). The prediction of the third feature is C_2. The forth feature value x_4 of e falls into the border of two partitions. In this case the representativeness values are used to determine the class value. Since the partition of class C_2 has a greater representativeness value than that of C_1 partition, the prediction of the forth feature is C_2. Final prediction of the CFP depends on the values of the feature weights (w_f's). If $w_1 > (w_3 + w_4)$ then CFP will classify e as a member of C_1; otherwise the prediction would be C_2.

The *sample complexity* and *training complexity* analysis of the CFP algorithm with respect to PAC-learning theory (Probably Approximately Correct learning) shows that, it requires small number of examples and a small amount of memory to learn a given concept, compared to many other similar algorithms.[3] Another outcome of this analysis is that, the CFP has a low learning complexity.

3. Empirical Evaluation

The performance of the CFP algorithm has been tested with various widely used real-world data sets. 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. All results in Table 1 were averaged

Table 1: Achieved accuracy (%) by CFP with various data sets

Database	CFP	Database	CFP
Cleveland	84.5	Mushroom	98.5
Hungarian	82.3	Thyroid Disease	90.8
Ionosphere	87.6	Voting	95.5
Iris	96.7	Wine	91.6

over 50 trials, unless indicated otherwise. The training and test sets were always disjoint. The instances were drawn randomly from the data sets. 80 % of the instances are used in training and the remaining are used in testing.

Ionosphere database: The radar data was collected by a system in Goose Bay, Labrador. The targets were free electrons in the ionosphere. *Good* radar returns are those showing evidence of some type of structure in the ionosphere. *Bad* returns are those that do not; their signals pass through the ionosphere. This data set contains 351 instances. 200 of the instances are used in training, the rest are used in testing. Each instance consist of 34 continuous-valued features. It is a binary (good or bad) classification problem.

Wine: The data are the results of a 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. Data set contains 178 instances and all attributes are continuous valued. The leave-one-out technique is used in the experiment.

Thyroid Disease: Five continuous-valued laboratory tests are used to predict a patient's thyroid type. The diagnosis was based on a complete medical record. There are 215 patient records in the data set each of which has one of the three classes (normal, hyperthyroid, or hypothyroid).

Mushroom: This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family. Each species is identified as edible, or poisonous. The database contains 8124 instances with 22 nominal-valued features. 1000 instances are used in training the remaining 7124 are used in testing.

Congressional Voting: It contains the voting records of the members of the United States House of Representatives during the second session of 1984. It is described by 16 boolean attributes and has 288 missing values among 435 instances. 350 of the voting records are used in training and remaining 85 are used in testing. The goal is to predict is the political party of a member given their voting record. The reported accuracies by IB1 and C4 on this data set, are 91.8 and 95.5, respectively[1].

Iris Flowers: Iris flowers data set consists of four integer-valued features and a particular species of iris out of three classes. The data set contains 150 instances. EACH[5] achieved 95.3 accuracy on this data using leave-one-out technique.

Medical Diagnosis: The Cleveland and Hungarian data sets contain heart disease diagnoses collected from the Cleveland Clinic Foundation and Hungarian Institute of Cardiology, respectively. Diagnoses are described by 13 numeric-valued features.

The objective is to determine whether a patient has a heart disease. The Cleveland data consists of 303 instances and the Hungarian data consists of 294 instances. Achieved accuracies by IB3 and C4 on Hungarian data, are 80.5 and 78.2 respectively. For Cleveland data, accuracies of IB3 and C4 are 78.0 and 75.5, respectively[1].

The feature weight adjustment rate and the generalization limits are domain dependent parameters of the CFP. In these experiments their values are determined (settings are given in Şirin (1993)) by trial and error, separately for each application domain. However, we noticed that the performance was not sensitive to the small changes in Δ and D_f settings.

4. Conclusion

The CFP algorithm is applicable to concepts, where each feature, independent of other features, can classify the concept. Usually, real-world data sets presents this behavior. Empirical results of the CFP on real-world data sets justify this claim.

This approach is a variant of algorithms that learn by projecting into one feature dimension at a time. 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 represent complex concepts. For domains, where concepts are nested or projection of the concepts are overlapping, performance of the CFP degrades.

One of the improvements of the CFP over other exemplar-based algorithms is its low classification complexity. Another important improvement is natural handling of unknown attribute values. Since the value of each attribute is handled separately, attributes with unknown values are simply ignored by the CFP.

The CFP uses feature weights to cope with irrelevant attributes. Determining the best values for the domain dependent parameters of the CFP is an optimization problem for a given domain. GA-CFP is a version of CFP that uses genetic algorithm to find a good setting of these parameters.[2]

References

1. D. W. Aha, D. Kibler and M. K. Albert. Instance–Based Learning Algorithms. *Machine Learning* **6** 37–66, 1991.

2. H. A. Güvenir and İ. Şirin. A Genetic Algorithm for Classification by Feature Partitioning. In *The Proceedings of ICGA '93*, (Illinois, 1993), pp. 543–548.

3. H. A. Güvenir and İ. Şirin. The Complexity of CFP: A Method for Classification Based on Feature Partitioning. (to appear) In *Lecture Notes on Artificial Intelligence*, Proceedings of the AI*IA'93, (Torino, 1993).

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

5. S. Salzberg, A Nearest Hyperrectangle Learning Method. *Machine Learning*, **6** 251-276, 1991.

6. İ. Şirin and H. A. Güvenir , *An Algorithm for Classification by Feature Partitioning.* Technical Report CIS-TR-9301, Bilkent University, Turkey, 1993.