Complexity of the CFP, a Method for Classification Based on Feature Partitioning

H. Altay Güvenir and İzzet Şirin

Computer Engineering and Information Science Department Bilkent University, Ankara 06533 TURKEY

Abstract. This paper presents a new methodology for learning from examples, called *Classification by Feature Partitioning* (CFP). 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 subdividing it into sub-partitions. It is shown that the CFP algorithm has a low sample and training complexity.

1 Introduction

Several representation techniques have been used to describe concepts for supervised learning tasks. Exemplar-based learning techniques store only specific examples that are representatives of other several similar instances. Previous implementations of this approach usually extended the nearest neighbor algorithm, which use some kind of similarity metric for classification. The classification complexity of such algorithms is proportional to the number of objects stored.

This paper presents another form of exemplar-based learning, called *Classification by Feature Partitioning* (CFP). The CFP 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, which is a representative subset of all training examples [1]. On the other hand, the CFP partitions each feature into segments corresponding to concepts. Therefore, a concept description learned by the CFP is a collection of feature partitions.

The CFP algorithm can be seen to produce a special kind of decision trees (e.g., ID3, [3]). Unlike ID3, the CFP probes each feature exactly once. An important difference between decision tree approach and the CFP is that the classification performance of the CFP does not depend critically on any small part of the model. In contrast, decision trees are much more susceptible to small alterations. Similar to CFP and ID3, the *probabilistic learning system* called PLS1, also creates orthogonal hyperrectangles by inserting boundaries parallel to instance space axes [4]. The PLS1 system starts from the most general description and applies only specializations.

The CFP algorithm is briefly described in the next section. Section 3 presents the *sample complexity* and the *training complexity* analysis of the CFP algorithm with respect to *Probably Approximately Correct* (PAC) learning theory [6]. The final section discusses the applicability of the CFP and concludes with a general evaluation of the algorithm.

2 The CFP Algorithm

The CFP algorithm learns the projection of the concepts over each feature dimension. An example is given as a vector of feature values plus a label that represents its class. A 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 representativeness value (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. 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: $< [-\infty, x_1], U, 0 >$, $< [x_1, x_1], C_1, 1 >$, $< [x_1, \infty], U, 0 >$; where U stands for *undetermined* partition. A partition can be extended towards a neighboring point of the same class in an undetermined partition. Assume that the second example e_2 with class C_1 is close to e_1 in feature f. In that case the CFP will generalize the partition at x_1 on f into an extended partition: $< [x_1, x_2], C_1, 2 >$.

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 new example, 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 of the partition is incremented by one. However, 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.

The training process of the CFP has two steps: (1) learning the feature weights, (2) learning feature partitions. In order to learn appropriate feature weights, 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 (all weights are initially set to 1).

Classification in 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 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 representation count is chosen. If e_f falls in a partition with undetermined 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

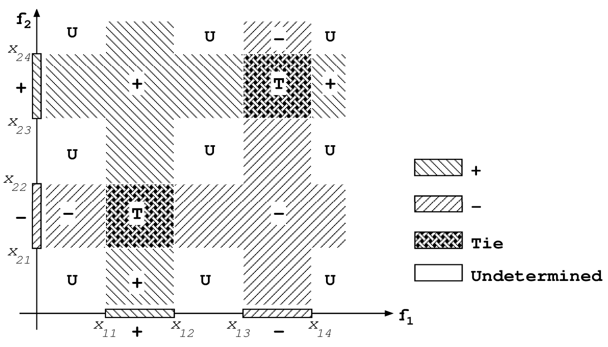

Fig. 1. An example concept description in a domain with two features.

that feature. The predicted class of a given instance is the one which receives the highest amount of votes among all feature predictions.

In order to illustrate the form of the concept descriptions, 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. 1. The corresponding concept description for the class (+) can be written 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 $(x_{23} \leq f_2 \;\&\; f_2 \leq x_{24} \;\&\; f_1 > x_{14})$

The CFP does not assign any classification to an instance, if it could not determine the appropriate class value. 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. If the features have different weights, the ties are broken in favor of the class predicted by the features with the highest weights during the voting process.

The CFP algorithm has been implemented and empirically evaluated in several standard domains and its performance has been compared with similar algorithms. In most of these domains the CFP algorithm attained higher accuracy than the other algorithms. The details of the CFP algorithm and the empirical comparisons can be found in [5].

3 Complexity of the CFP

This section presents an analysis of the CFP algorithm with respect to *PAC-learning* theory [6]. 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.

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 learnability, but modifies it for the voting based classification used in the CFP. We first determine the minimum number of training instances required 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 1. 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 $|s_f - x_f| < \varepsilon$ at least for d values of f ($1 \leq f \leq n$).

Lemma 2. *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 $m > \lceil 1/\varepsilon \rceil / \gamma \times (nln2 + ln(\lceil 1/\varepsilon \rceil / \delta))$.

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. □

[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 r objects out of n.

Theorem 3. *Let ε, δ, and γ be fixed positive numbers less than one and S a sample set with n features. If $\lceil\frac{n+1}{2}\rceil$ 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. Following the proof of the lemma, if S form an $<\varepsilon, \gamma, d>$-net, then it is guaranteed 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. $\square$

Theorem 4. *Let ε, δ, and γ be fixed positive numbers less than one. If a random sample set S with n features forms an $<\varepsilon, \gamma, \lceil\frac{n+1}{2}\rceil>$-net with confidence greater than $1-\delta$, then CFP with $D_f \geq 2\varepsilon$ 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 $D_f \geq 2\varepsilon$) partition for each sub-interval. Thus, for n features, the CFP constructs at most $n\lceil 1/\varepsilon\rceil$ partitions. $\square$

Theorem 5. *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 $D_f \geq 2\varepsilon$ 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 $O(nlog(\lceil 1/\varepsilon\rceil))$ for n features. Consequently, with m training instances, the training complexity of the CFP algorithm is $O(mn\log(\lceil 1/\varepsilon\rceil))$. $\square$

The classification process in exemplar-based learning algorithms which use some form of the nearest neighbor algorithm 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 $O(n\log M)$, which significantly reduces the classification complexity.

4 Conclusion

The CFP algorithm is applicable to concepts, where each feature, independent of the others, can be used to classify the concept. 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 find complex concepts.

The analysis of the CFP 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. Another outcome of the analysis is that, the CFP has also a low training complexity.

The real-world data sets usually contain missing attribute values. Most of the learning systems usually overcome this problem by either filling in missing attribute values, or looking at the probability distribution of values of attributes. In contrast, the CFP solves this problem very naturally. Since the CFP treats each attribute value separately, in the case of an unknown attribute value, it simply leaves the partitioning of that feature intact.

The CFP uses feature weights to cope with irrelevant attributes. Introducing feature weights protects the algorithm's performance, when attributes have different relevances. 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 D_f generalization limit for each attribute, which controls the generalization process. Δ and D_f 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. A version of CFP, called GA-CFP, has been implemented to learn these parameters using genetic algorithms [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, *Proceedings of the ICGA'93*, Illinois, 1993.
3. J. R. Quinlan, Inductions of Decision Trees. *Machine Learning* **1**, 81-106, 1986.
4. L. Rendell and H. Cho, Empirical Learning as a function of Concept Character, *Machine Learning* **5** 267–298, 1990.
5. İ Şirin and H. A. Güvenir, *An Algorithm for Classification by Feature Partitioning.* Technical Report CIS-9301, Bilkent University, Dept. of Computer Engineering and Information Science, Ankara, 1993.
6. L.G. Valiant, A Theory of the Learnable. *Communications of the ACM*, **27** (11) 1134-1142, 1984.

This article was processed using the LaTeX macro package with LLNCS style