Learning Symbolic Prototypes

Piew Datta and Dennis Kibler
Department of Information and Computer Science
University of California
Irvine, CA 92717
{pdatta, kibler}@ics.uci.edu

Abstract

We present an empirical analysis of symbolic prototype learners for synthetic and real domains. The prototypes are learned by modifying the minimum-distance classifier to solve problems with symbolic attributes, attribute weighting, and its inability to learn multiple prototypes for a class. These extensions are implemented in SNMC. In the second half of this paper, we provide empirical analysis, characterizing situations where symbolic prototypes have advantages over traditional methods such as decision trees and instance-based methods. Empirical analysis on real domains show that SNMC increases classification accuracy by 10% over the original minimum-distance classifier and has a higher average generalization accuracy than both C4.5 and PEBLS on 20 domains from the UCI data repository. Finding multiple prototypes for classes results in the same or higher accuracy than learning a single prototype for classes.

Keywords: Inductive learning, classification, instance-based learning, prototype learning, clustering, experimental evaluation.

Email: pdatta@ics.uci.edu

Phone number: (714) 824-2111

Submission statement: A portion of this paper overlaps another paper submitted to another conference. If both papers are accepted the overlap will be reduced by referencing the other paper and additional research will be added. The overlap was necessary to explain some of the ideas.

Learning Symbolic Prototypes

Abstract: We present an empirical analysis of symbolic prototype learners for synthetic and real domains. The prototypes are learned by modifying the minimum-distance classifier to solve problems with symbolic attributes, attribute weighting, and its inability to learn multiple prototypes for a class. These extensions are implemented in SNMC. In the second half of this paper, we provide empirical analysis, characterizing situations where symbolic prototypes have advantages over traditional methods such as decision trees and instance-based methods. Empirical analysis on real domains show that SNMC increases classification accuracy by 10% over the original minimum-distance classifier and has a higher average generalization accuracy than both C4.5 and PEBLS on 20 domains from the UCI data repository. Finding multiple prototypes for classes results in the same or higher accuracy than learning a single prototype for classes.

1 Introduction

Two types of representations have been used to describe concepts: representations that include only those attributes necessary to discriminate one concept from another (e.g. decision trees, rules) and those that characterize the concept. Algorithms using only discriminating attributes (Quinlan, 1993; Brieman et. al., 1984; Clark & Boswell,1991) tend to learn the minimal description necessary to discriminate classes. Instance-based algorithms (Aha, Kibler & Albert, 1991; Cost & Salzberg, 1993) represent all of the attributes for the concept but use all training examples as the concept description, revealing little intuition about the concepts.

Some researchers (Duda & Hart, 1973; Aha, Kibler & Albert, 1991; Zhang, 1992; Skalak, 1994; Datta & Kibler, 1995) have stored only a subset of the examples or a generalization of the examples. Duda & Hart (1973) discuss a variation of the nearest neighbor classifier termed the minimum-distance classifier. This classifier works similarly to the nearest neighbor classifier except that the mean for each class is stored as a prototype. During classification time, the class of closest prototype is predicted.

The minimum-distance classifier has some drawbacks. It cannot use symbolic (nominal) attributes since the mean for symbolic attributes and the distance between two attribute values is not pre-defined. Another drawback is its inability to weigh attributes. Attributes not relevant for classification

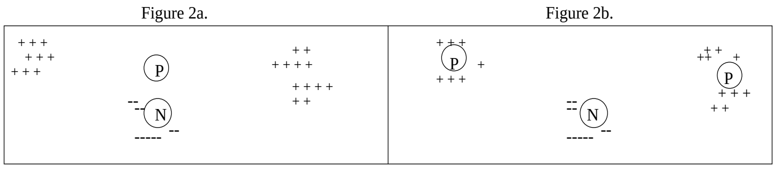

Figure 1: + and − class shown. P and N denote prototypes for the + and − classes respectively. 1a: The single prototype P is insufficient to represent +. 1b: Multiple prototypes better represent + examples.

may distract the classifier. The last drawback is its inability to learn more than one prototype for each class. Using the mean of a class as a prototype will not work well if the prototype is actually very distant from the examples as illustrated in Figure 1a. If the minimum-distance classifier could learn multiple prototypes (Figure 1b.) then it will classify better in domains where a single prototype per class is not sufficient.

This paper has three goals: identify problems with the minimum-distance classifier, propose solutions, and characterize situations where our solutions perform better than traditional machine learning methods such as C4.5 and PEBLS. Section 2 introduces the mean for symbolic attributes. Section 3 shows how to weigh attributes and introduces an algorithm SNM (Symbolic Nearest Mean) that uses symbolic attributes and attribute weighting. Section 4 describes an extension, SNMC, to the minimum-distance classifier for learning multiple prototypes for classes. In section 5 we provide experimental evaluations for SNM and SNMC in synthetic domains and in 20 domains from the UCI Repository. We use synthetic domains to characterize situations where SNMC, C4.5 and PEBLS excel. We also compare SNM and SNMC with C4.5 (Quinlan, 1993), PEBLS (Cost & Salzberg, 1993), and the minimum-distance classifier using generalization accuracy as our comparison metric. Section 6 summarizes our contributions.

2 Symbolic Attributes

The distance metric for nearest neighbor and minimum-distance classifier is crucial to their predictive capabilities. Euclidean distance, a commonly used metric, is defined as

$$E(x,y) = \sqrt{\sum_{i=1}^{a} d(x_i,y_i)^2}$$

where x and y are two examples, a is the number of attributes and x_i refers to the ith attribute value for example x. For real-value attributes $d(x_i, y_i)$ is defined as their absolute difference (ie. $|x_i - y_i|$).

For symbolic attributes the distance between two values is typically defined as $d(x_i, y_i) = 0$ iff $x_i = y_i$ and 1 otherwise. This assumes that each different symbolic value is equi-distant from another which leads to problems when two different values should be considered equal, and where symbolic values should have varying distances among them. We propose a different distance metric for symbolic attributes and generalize the method for determining the mean of a set of values.

2.1 Distances between Values

Stanfill & Waltz (1986) developed the Value Distance Metric (VDM) for computing different distances among symbolic values. The VDM has two components, attribute weights (denoted W) and distances between two values (denoted D). We only use the D portion of the VDM which is similar to MVDM (Cost & Salzberg, 1991) since the W component is not symmetric. MVDM is based on the ratio of the probability of the value occurring in each class. $D(v, w)$ is defined by

$$D(v, w) = \sum_{i=1}^{C} \left| \frac{f(C_i, S_v)}{|S_v|} - \frac{f(C_i, S_w)}{|S_w|} \right|$$

where v and w are two values from an attribute, C is the number of classes, C_i is the ith class, S_v is the set of examples that have the value v for attribute A, $f(C_i, S_v)$ is the number of examples with class C_i in the set S_v, and $|S_v|$ is the number of examples in S_v. Consequently, the calculated distances between symbolic values are normalized between 0 and 1 (after dividing by the number of classes). We also normalize the real-values by dividing by their range in the training set. Thus the distances between real-value and symbolic attributes are relative and comparable. All of the distances between symbolic attribute values are calculated once for the training set and stored in the VDM table before learning takes place.

2.2 Calculating the Mean

The simplest method for determining the mean of a symbolic attribute is to choose the most common value; however, this does not yield the best classification accuracy. In addition, the mean of a symbolic attribute must be one of its possible values. For numeric data the mean is the value that minimizes the variance. We generalize this notion to deal with symbolic attributes as well. We define the mean of a set of symbolic values by finding the value of μ minimizing the variance, that is

$$\mu = argmin(\sum_{v \in J} D(v, \mu)),$$

where J is the set of values and v is a symbolic value in J. μ will act as the best constant approximation for the symbolic values in J similar to the mean for real values. Computationally, each symbolic value will be tried as μ and the symbolic value that minimizes the variance will become the mean for J. We define the mean of a set of examples, S, to be the vector $< A1_\mu, A2_\mu, ...An_\mu >$ where Ai_μ denotes the mean of the ith attribute. We call this vector the *prototype* for S.

We have developed an algorithm called SNM (Symbolic Nearest Mean) that has a more robust method for determining distances between symbolic values. SNM uses the MVDM and the definition of mean described above. SNM learns a prototype for each class, classifies examples by finding the closest prototype using Euclidean distance, and predicts the prototype's class.

3 Weighing Attributes for Classification

Inferring the applicability of an attribute is important for the classification task. The minimum-distance classifier weighs all of the attributes equally which may decrease classification accuracy. The MVDM provides a weighing method for symbolic attributes based on classification relevance. For example, if the distances between the values for symbolic attribute A range from .01 to .02 then this attribute has less weight than symbolic attribute B which has distances varying from 0.7 to 0.9 . The MVDM provides a more precise weighing method than simply stating that an attribute has a particular weight; it weighs each value of the symbolic attributes.

The MVDM has some similar characteristics to the information gain metric (Quinlan, 1993) and the ORT metric (Fayyad & Irani, 1992).

Lemma 1 *If the MVDM for attribute A = 0 (i.e. D(v,w)=0 $\forall$ values k of A) then the information gain for A is 0.*

Proof: Let $p_i = \frac{f(C_i,S)}{|S|} \equiv$ the number of elements of class i in S divided by the number of elements in S. Let $p_{ij} = \frac{f(C_i,S_j)}{|S_j|}$. If S is separated into j subsets, then in general $p_i = \sum_j \frac{|S_j|}{|S|} p_{ij}$. Since the MVDM for A = 0, then

$p_{ij} = p_{ik}\ \forall j,k$. Thus $p_i = \sum_{j=1}^{k} \frac{|S_k|}{|S|} p_{ij} = p_{ij}$. Therefore $p_i v = p_i \forall v$. Using this information in the definition of information gain, we have

$$infogain_A(S) = \left(-\sum_{i=1}^{C} \frac{f(C_i,S)}{|S|} log_2 \frac{f(C_i,S)}{|S|} \right) - \left[\sum_{j=1}^{k} \frac{|S_j|}{|S|} \left(-\sum_{i=1}^{C} \frac{f(C_i,S_j)}{|S_j|} log_2 \frac{f(C_i,S_j)}{|S_j|} \right) \right].$$

$$infogain_A(S) = \left(-\sum_{i=1}^{C} p_i\, log_2\, p_i \right) - \sum_{j=1}^{k} \frac{|S_j|}{|S|} \left(-\sum_{i=1}^{C} p_{ij}\, log_2\, p_{ij} \right).$$

$$infogain_A(S) = \left(-\sum_{i=1}^{C} p_i\, log_2\, p_i \right) - \sum_{j=1}^{k} \frac{|S_j|}{|S|} \left(-\sum_{i=1}^{C} p_i\, log_2\, p_i \right).$$

$$infogain_A(S) = \left(-\sum_{i=1}^{C} p_i\, log_2\, p_i \right) - 1 \left(-\sum_{i=1}^{C} p_i\, log_2\, p_i \right).$$

$$infogain_A(S) = 0.$$

If the MVDM of A is 0 then partitioning the set of examples S according to A will not change the class probabilities of the resulting sets S_j. The ORT metric also compares the class probabilities of the resulting sets. If the class probabilities of the resulting sets are equal then the ORT metric is minimized (Fayyad & Irani, 1992) like the MVDM and the information gain metric.

Thus far SNM does not weigh real-value attributes. A simple method of weighing real-value attributes is to discretize them and use the MVDM to define the distances between each of their new discretized values. It may seem unintuitive to discretize real-value attributes since nearest neighbor algorithms are naturally suited to real-value attributes. However, these distances do not necessarily reflect the usefulness of the values for classification. For example, suppose we have a real-value attribute denoting the average number of cigarettes a person smokes daily. Intuitively there is a large difference between person A who smokes 0 cigarettes and person B who smokes 1 cigarette although the actual difference between the number of cigarettes is 1. The difference between the number of cigarettes person C smokes (say 10) and person D (say 11) is also 1. Although the actual difference between each pair of these cigarette smokers (A & B versus C & D) is 1 , for classification purposes we want a large difference between persons A and B and a small difference between persons C and D.

3.1 Evaluating Weighting and Discretization

To test our hypothesis that discretizing real-value attributes would increase the classification accuracy of SNM, we compared its classification performance with and without discretization. We applied a discretization process that orders the attribute values and creates intervals for the corresponding symbolic values such that the intervals contain an approximately equal number of examples. Results in Dougherty, Kohavi, & Sahami (1995) suggest that 10 intervals are satisfactory and this agrees with our experiments. Intuitively, if we create too many intervals (more than necessary for maximum accuracy), then the MVDM will tend to give zero distance to intervals that should be merged. If too few intervals are created, however, the MVDM has no way of separating the intervals.

Unless otherwise specified all experiments average 30 runs of randomly choosing (without replacement) two-thirds of the data as a training set and the remainder as the test set. Both sets are stratified. We realize that the test sets in the 30 runs are not independent. Kohavi(1995) recommends multiple runs of cross validation, but we contend that this methodology suffers from the same independence problem. Table 1 shows the results on 11 domains from the UCI Data Repository (Murphy & Aha 1994) containing either a mixture of symbolic and real-value attributes or only real-value attributes. The table shows that in almost all domains, (except the Breast Cancer Wisconsin domain which appears to be discretized in its original form), discretization aids in classification accuracy or does not decrease accuracy significantly. By weighing the real-value attributes through the MVDM, SNM now has a method of choosing the more relevant attributes for classification. Therefore, we will adopt the discretization of real-value attributes in SNM.

4 Learning Prototypes Via Clustering

The last drawback of the minimum-distance classifier is that it works best on domains where there is one prototype for each class. The classification accuracy of SNM and the minimum-distance classifier decreases as more classes require multiple prototypes to represent the examples. Learning more than one prototype per class would allow SNM to represent distant and disjoint groups of examples within the classes. The difficulty lies in determining the grouping of examples for the prototypes.

One possible approach to finding groups of examples in the same class is clustering which is typically used in an unsupervised task for data explo-

Table 1: Accuracy of SNM with and without discretization (* indicates a statistical significance at the 95% level or higher using a two-tailed t-test)

Domain	SNM w/o Discr.	SNM w/ Discr.
Breast Can.	66.0%	69.9%*
Breast C. Wis.	95.7%*	91.6%
Credit	84.8%	85.0%
Flag	55.7%	69.5%*
Glass	43.9%	60.2%*
Cl. Hrt. Dis.	83.6%	83.2%
Hepatitis	78.6%	81.9%*
Iris	91.5%	91.4%
Pima Ind. Dia.	72.8%	72.9%
Waveform	80.5%	81.6%*
Wine	95.2%	95.8%

ration. In this task the examples are not presorted into classes and there is no clear metric to optimize. In our similar supervised classification task we want to find clusters within each of the classes. The next section describes SNMC (Symbolic Nearest Mean with Clustering), our modification to SNM for representing multiple prototypes within classes. First we describe k-means clustering, a method for determining the clusters, and then describe a method of deciding on k, the number of clusters for each class.

4.1 K-Means Clustering

A common algorithm, k-means clustering (Duda & Hart, 1973; MacQueen, 1967), uses a distance metric to place examples into k clusters. The algorithm continues moving examples to their closest cluster (the cluster mean) until no more examples are moved. The two parameters to this algorithm are the distance metric and k, the number of clusters to create. Euclidean distance is the most commonly applied distance measure, although other distance metrics exist such as city block distance and Mahalanobis distance (Duda & Hart 1973). Finding k, on the other hand, is a more serious open problem. Duda & Hart, Smyth(1996), Cheeseman & Stutz(1996) and other researchers have suggested various methods to determine the number of clusters in a group of examples.

For each class c set $k_c = 1$.
Repeat
  For each class c
    If ($k_c + 1$ *improves* accuracy) then learn one more cluster for c.
Until (accuracy has not improved)
Learn prototypes for each cluster.

Figure 2: Pseudocode for SNMC.

4.2 Finding k through Training Set Accuracy

Although class information is not helpful in creating the clusters (since all examples will be from the same class), it can help to restrict the number of clusters to create. SNMC uses the accuracy on the training set as a feedback mechanism for learning a vector k where k_c is the number of clusters for class c. The general idea behind SNMC is to create more clusters in a particular class if it increases the classification accuracy of the prototypes representing the training examples. The bias in SNMC is to create as few clusters as possible, but to increase k_c when it leads to an increase in accuracy.

Figure 2 shows the procedure for SNMC. SNMC uses hill-climbing to find k_c. More specifically, it temporarily increments the number of clusters in each class to see if the classification accuracy on the training set has increased. k_c is only incremented if it results in an increase in classification accuracy on the training set. Overall, SNMC separates the training examples by class, attempts to find clusters that maximize classification accuracy on the training examples[1] and generalizes the clusters into prototypes. SNMC predicts classes for unseen test examples by finding the closest prototype and predicting the class of the prototype.

[1] SNMC calls k-means 5 times with the same k_c and chooses the cluster with the highest classification accuracy to represent the accuracy of k_c. This helps to overcome the dependency k-means has on an initial random state.

5 Empirical Evaluation

We describe two methodologies applied to evaluate SNM and SNMC: synthetic domains and real world domains. The synthetic domains allow control of various characteristics of the domains, such as the type of concept, the number of attributes and examples, the amount of noise and irrelevant attributes. Real world domains provide feedback on whether the algorithms can be applied to practical problems.

5.1 Synthetic Domains

We experimented with two types of concepts that appear useful in many actual domains. The first type requires necessary conditions to be true. Typically this concept can be described by small conjunctions and disjunctions of attributes. We call these concepts *crisp* . Decision tree and rule learners can find this type of concept since they have the appropriate representation heuristic, and search bias. The second type of concept is a *k-of-n* concept where a number k of the n attributes need to be *on* for class membership. In these concepts, there is no subset of attributes relevant for predicting class membership, but rather each attribute contributes a small amount of evidence. If the amount of evidence is above a threshold then class membership is predicted. An example of this type of domain is disease prediction where a patient does not need to have any particular symptom to have the disease, but should have a number of symptoms. We call these concepts *cumulative.*

5.1.1 Crisp Concepts

We created crisp concepts and varied three characteristics: the number of disjunctions, and the class and attribute noise. The domain was created by using 10 binary attributes (denoted A...J) and 2 classes (denoted C1 C2). Four relevant attributes and values were randomly chosen for the concepts (eg. A=1,D=0, E=1,F=0) that describe the necessary conditions for class membership to a random class (eg. C1←(A=1 & D=0 & E=1 & F=0)). Values for the irrelevant attributes were chosen randomly.

In the first experiment we varied the number of disjunctions from 0 disjuncts (1 conjunction) to 4 disjuncts in each class. Figure 3a. shows the result of C4.5, PEBLS, SNM, and SNMC on this domain. Each data point is the result of creating 30 synthetic concepts with about 400 examples each. C4.5 is suited to this type of domain and can classify better than the other

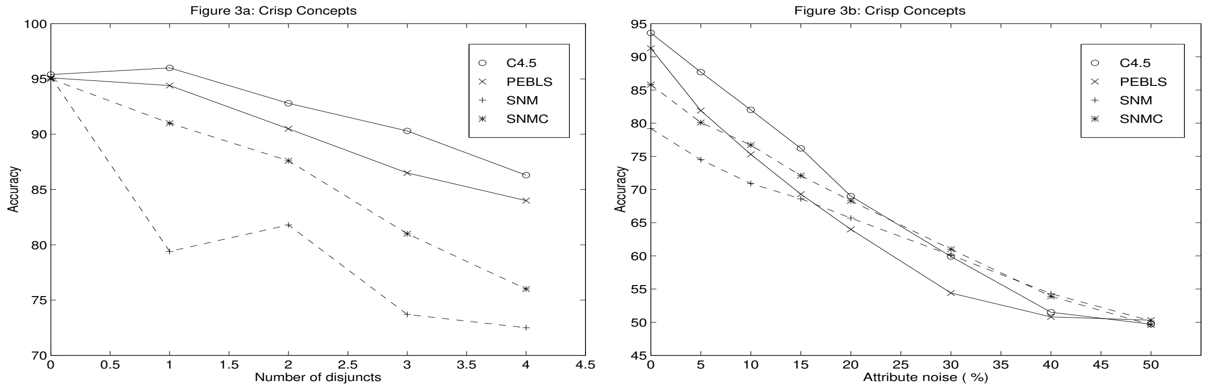

Figure 3: Crisp concepts. 3a: Accuracy vs. number of disjuncts. 3b: Accuracy vs. attribute noise.

algorithms. SNMC performs better than SNM since it can learn multiple prototypes per class, however, its prototype representation is not as appropriate as a decision tree representation for these types of concepts.

In the second experiment we added attribute noise to the concepts where each attribute value has an e% chance of being incorrect. We varied e from 0% to 50% and had 1-2 disjunctions in the concepts. In Figure 3b. C4.5 performs the best with small amounts of noise, although SNMC performs close to C4.5 with larger amounts of noise. With greater than 10% attribute noise SNMC performs better than PEBLS, showing that SNMC is more resilient to attribute noise than PEBLS. We also conducted this same experiment with class noise where the class attribute has an e% chance of being incorrect. We varied the class noise from 0% to 50% and obtained very similar results to the attribute noise experiment. As expected, when concepts are crisp, discriminating learners perform better than characteristic learners; however, the difference between the two learners capabilities decreases with increasing noise since SNMC performs similar to C4.5 and better than PEBLS.

5.1.2 Cumulative Concepts

Cumulative (k-of-n) concepts consist of n binary attributes and two classes. If at least k of the attributes are *on* then the example is a member of the positive class. We used n=8 and created all 256 examples. We varied k from 3 to 6 to see affects on classification accuracy (Figure 4a.). Choosing the most common class for k=3 to 6 results in accuracies of 85%, 58%, 63% and 85% respectively. SNMC is the only algorithm that consistently performs well. SNM performs the opposite of C4.5 and PEBLS. There are no attributes

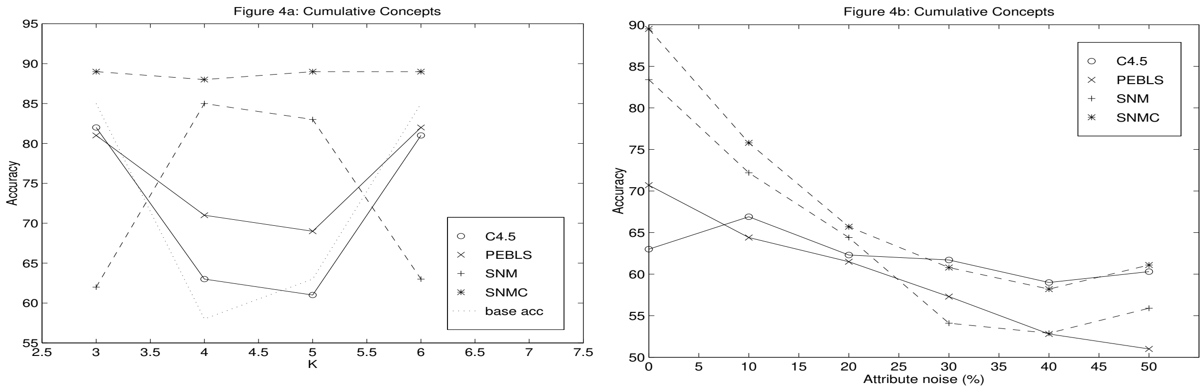

Figure 4: Cumulative concepts. 4a: Accuracy vs. K Figure 4b: Accuracy vs. attribute noise.

in this domain that have high information gain. C4.5 learns large trees and tends to overfit the training sample. Overfitting results in better classification accuracy when class sizes are very uneven (k=3 and 6) rather than when class sizes are more equal (k=4 and 5). For most values of k, C4.5's accuracy is less than the base accuracy. PEBLS's classification accuracy is similar to C4.5. PEBLS also classifies more accurately when there is a large difference between class sizes (eg. k=3 and 6), although it is worse than using the base accuracy. When k=4 and 5, the class sizes are more similar and PEBLS can distinguish between the classes better than merely using base accuracy.

To view the affects of noise, we added attribute noise which varied from 0% to 50% while the number of attributes (n=8) and k (k=4) remained constant. Figure 4b. shows the results from adding attribute noise. At high levels of noise ($> 30\%$) C4.5 and SNMC perform about the same. Similar results occur when varying the class noise from 0% to 50%, although C4.5 never classifies more accurately than SNMC. In both noise experiments, C4.5 has a slight increase in accuracy at 10% noise compared to 0% noise. When no noise is present, C4.5 tends to overfit the training data, creating fairly large decision trees, however with small amounts of noise (eg. 10%) the overfitting is reduced resulting in smaller decision trees and a slight gain in accuracy. As in the necessary conditions domain, PEBLS' accuracy declines quickly after small amounts of either type of noise. The last experiment with cumulative concepts explores the effects of irrelevant attributes. We added 0 to 4 irrelevant attributes (with random values) holding n at 8 and k at 4. The irrelevant attributes did not affect any of the four algorithms.

For learning pure cumulative concepts, SNMC performs the best of the four algorithms. When noise is added SNMC performs the best, C4.5 classifies almost as well as SNMC, and PEBLS is greatly handicapped. C4.5 is biased towards crisp concepts, since it searches for a few relevant attributes. A prototype representation is more appropriate for cumulative concepts.

5.2 Real World Domains

We compared C4.5, PEBLS, the minimum-distance classifier, SNM, and SNMC on 20 domains from the UCI Repository (Murphy & Aha, 1994). Three of these domains are artificial (LED domains and Hayes Roth) and the remaining are real world domains. They have a variety of characteristics including different types of attributes, and a large range in the number of examples, attributes, and classes. For the Waveform domain we used a 50/50 split of the examples for training and test sets. These results are shown in Table 2. The table represents the accuracy ± the standard deviation followed by the rank of the algorithm; the algorithm with the highest accuracy has rank 1. The average accuracy and average rank of each of the algorithms are shown at the bottom of the table. The average number of prototypes per class that SNMC learns is shown in the last column.

SNMC has the highest average accuracy over these domains with PEBLS and C4.5 coming in second and third respectively. SNMC has an average increase in accuracy of about 2% over C4.5 and PEBLS. SNMC also has the best rank of these algorithms (2.1) followed by PEBLS and SNM. In fact, considering the simplicity and efficiency of SNM, it classifies surprisingly well. Similar results have been seen with other simple methods (Auer, Holte, & Maass, 1995; Iba & Langley, 1992). The minimum-distance classifier has the lowest average accuracy and rank of all the algorithms. The proposed solutions for the three problems described in this paper result in almost a 10% increase in accuracy from the minimum-distance classifier to SNMC.

SNMC's average number of prototypes per class is fairly small. For domains where the concepts are known such as the LED domains, SNMC learns 1.1 prototypes per class which is close to the correct number of prototypes. In the Iris domain, even though there are 3 classes, one of the classes contains a disjunction. SNMC also learns the correct number of prototypes per class (1.3) for this domain. Also the number of prototypes SNMC learns is not correlated with the number of examples in the domains ($r = -0.24$).

When comparing SNMC with C4.5 and PEBLS, SNMC has a higher accuracy than C4.5 in 13 of the 20 domains and has a higher accuracy than PEBLS in 12 of the 20 domains. Applying the Wilcoxon test, the null hypothesis that C4.5 and SNMC have equal accuracies is rejected at the 95%

Table 2: Accuracy and Number of Prototypes per Class for SNMC

Domain	**C4.5**	**PEBLS**	**Min-dis.**	**SNM**	**SNMC**	**pt/cl**
1. Audiology	88.2±4 (2)	89.4±3 (1)	72.8±6 (5)	82.4±5 (3)	81.9±4 (4)	1.4
2. Breast Can.	73.0±3 (1)	67.3±4 (4)	62.0±8 (5)	70.2±5 (3)	72.3±3 (2)	1.3
3. Br. C. Wis.	95.1±1 (2)	92.6±2 (4)	95.8±1 (1)	91.8±1 (5)	94.2±2 (3)	1.3
4. Cl. Hrt. Dis.	71.9±6 (5)	77.1±3 (4)	80.5±4 (3)	83.5±3 (1)	83.3±3 (2)	1.1
5. Credit	84.4±2 (3)	82.2±2 (5)	83.1±2 (4)	85.2±2 (2)	85.3±2 (1)	1.1
6. Flag	72.5±5 (1)	65.3±5 (4)	55.0±6 (5)	67.4±5 (2)	65.7±5 (3)	1.8
7. Glass	68.4±6 (2)	70.3±5 (1)	41.0±7 (5)	61.1±7 (4)	67.7±6 (3)	1.8
8. Hayes Roth	81.2±4 (3)	85.5±4 (1)	75.4±5 (4)	38.4±7 (5)	83.2±6 (2)	2.5
9. Hepatitis	80.3±5 (3)	79.3±6 (4)	77.0±6 (5)	82.9±5 (1)	82.5±5 (2)	1.5
10. House Vt.	96.1±1 (1)	94.2±1 (3)	87.3±3 (5)	90.4±2 (4)	94.9±2 (2)	2.9
11. Iris	95.4±3 (1)	93.7±4 (2.5)	92.1±4 (5)	92.7±4 (4)	93.7±4 (2.5)	1.3
12. LED 200 exs	57.2±5 (4)	48.3±5 (5)	62.7±13 (3)	67.6±5 (1)	65.1±5 (2)	1.2
13. LED-24	67.3±4 (3)	62.8±4 (4)	29.7±4 (5)	73.0±3 (1)	72.5±3 (2)	1.1
14. Lymph.	79.5±5 (4)	82.9±4 (1)	71.0±7 (5)	82.1±4 (3)	82.5±5 (2)	1.4
15. Pima I. D.	72.1±3 (4)	69.4±3 (5)	73.1±2 (3)	75.3±3 (2)	75.8±3 (1)	1.4
16. Promoters	78.3±7 (4)	89.0±4 (3)	71.3±9 (5)	91.4±5 (1)	90.6±5 (2)	1.8
17. Soybean-lg	86.8±4 (3)	92.4±2 (1)	68.4±5 (5)	82.2±4 (4)	86.9±4 (2)	1.3
18. Waveform	74.9±.9 (5)	77.7±.9 (4)	79.2±6 (3)	81.1±.8 (2)	81.4±1 (1)	1.0
19. Wine	92.7±4 (5)	97.4±2 (1)	95.6±2 (4)	96.1±3 (3)	97.1±2 (2)	1.3
20. Zoo	98.4±2 (4)	100±0 (1)	98.3±3 (5)	99.5±2 (3)	99.7±2 (2)	1.0
Ave. Accuracy	80.7	80.8	73.6	79.7	82.8	n/a
Ave. Rank	3	2.9	4.25	2.7	2.1	n/a

significance level. When applying the Wilcoxon test to PEBLS and SNMC, the null hypothesis is rejected at the 95% significance level. According to the Wilcoxon test, SNMC is significantly different than C4.5 and PEBLS and the average accuracy of SNMC shows it is more accurate than C4.5 and PEBLS over these 20 domains.

Figure 5a shows the difference in the accuracies between SNMC and C4.5 on the 20 domains (ordered by increasing difference). The x-axis shows the domain number (domain numbers are in Table 3) and the y-axis shows their difference. The poor performance of SNMC in the Audiology domain can be explained by SNMC's replacement of missing values by their most common value in the class. Treating missing values as another valid value in this domain boosts SNM and SNMC's accuracies by 3-4%. In domains 11, 10,

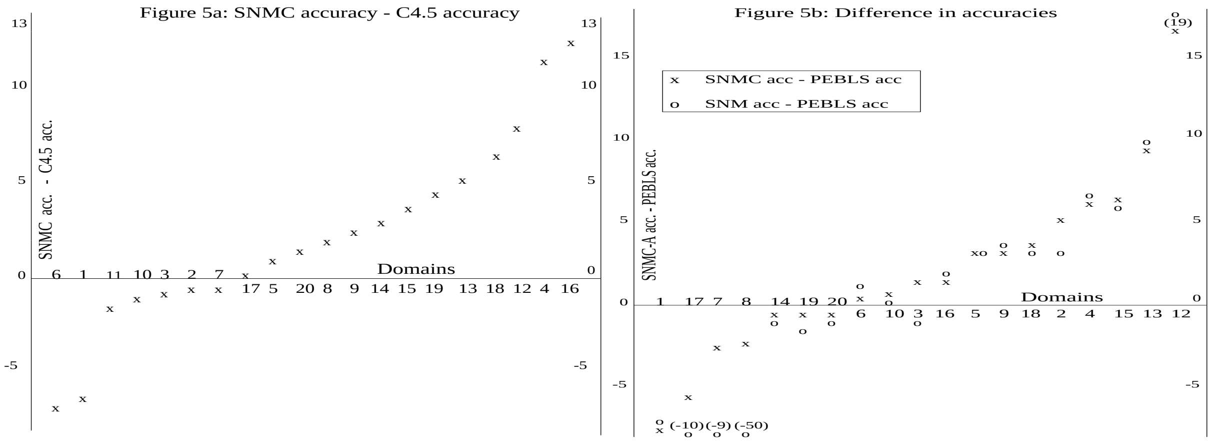

Figure 5a: Difference between SNMC accuracy and C4.5 accuracy. Figure 5b: Difference between SNM, SNMC, and PEBLS accuracies.

3, 2, 7, 17, and 5 SNMC and C4.5 can be considered similar and in the remaining domains (right of 5) SNMC is better than C4.5.

PEBLS can be considered a special case of SNMC where each k_c is exactly the number of training examples in each class c. The three algorithms (SNM, SNMC, and PEBLS) provide a spectrum of possible values for k_c. Figure 5b. shows the difference between SNMC's accuracy and PEBLS' accuracy (denoted by x) and the difference between SNM's accuracy and PEBLS'accuracy (denoted by o) for the 20 domains. The domains are ordered by increasing difference between SNMC and PEBLS. In the domains where SNMC classifies worse than PEBLS (left of 14), SNM classifies much worse than SNMC. This implies that these domains contain more than one prototype per class and SNMC does not learn enough prototypes for these domains. The domains where SNMC classifies better than PEBLS (right of 10) implies that SNMC learns a more appropriate number of prototypes per class than PEBLS or that the domains contain noise (eg. LED and LED-24) which greatly degrades PEBLS accuracy (shown in synthetic domains). SNMC typically classifies better than SNM (domains 17, 7, 8, 10, 3, 2) or very similar to it.

In an attempt to increase SNMC's accuracy in those domains where SNMC classifies more poorly than PEBLS, we weakened SNMC's bias for fewer prototypes. Originally SNMC will not increase k_c unless the training set accuracy increases. We modified SNMC so that it increases k_c when the accuracy increases or does not change. We ran this modification on the same UCI domains and got an average accuracy of 82.1% and 4.5 as the average number of prototypes per class. This is a small change from the original SNMC's statistics (recall the original SNMC's average accuracy of 82.8 and 1.5 prototypes on average). Overall this modification did not improve SNMC's accuracy.

6 Contributions

This paper identifies three problems with the minimum-distance classifier: its inability to work with symbolic attributes, weigh attributes, and learn disjunctions within classes. We proposed SNM and SNMC to tackle these problems. Our empirical evaluation on synthetic domains implies that SNMC classifies the best of the algorithms tested on cumulative concepts. In addition these results show that SNMC is more resilient to attribute and class noise than PEBLS in both crisp and cumulative concepts. The experimental results on the UCI domains shows that SNMC classifies the best (both in rank and accuracy) of the five algorithms.

References

Aha, D., Kibler, D. & Albert M. (1991). Instance-based learning algorithms. *Machine learning*, volume 6, pp 37-66. Boston, MA: Kluwer Publishers.

Auer, P., Holte, R. & Maass, W. (1995). Theory and applications of agnostic PAC-learning with small decision trees. In *Proceedings of the Twelfth International Conference on Machine Learning*. Tahoe City, CA. Morgan Kaufmann.

Brieman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and Regression Trees.* Monterey, Ca. Wadsworth & Brooks/Cole Advanced Books & Software.

Cheeseman, P. & Stutz, P. (1996). Bayesian classification (AutoClass): theory and results, in *Advances in Knowledge Discovery and Data Mining*, U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.), Cambridge, MA: AAAI/MIT Press, pp. 153-180.

Clark, P. and Boswell, R. (1991). Rule Induction with CN2: some recent improvements. *European Working Session on Learning*.

Cost, S. and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, volume 10, pp. 57-78. Boston, MA: Kluwer Publishers.

Datta, P. and Kibler, D. (1995). Learning Prototypical Concept Descriptions. In *Proceedings of the Twelfth International Conference on Machine*

Learning. Tahoe City, CA. Morgan Kaufmann.

Dougherty, J., Kohavi, R., Sahami, M. (1995). Supervised and Unsupervised Discretization of Continuous Features. In *Proceedings of the Twelfth International Conference on Machine Learning.* Tahoe City, CA. Morgan Kaufmann.

Duda, R., and Hart P. (1973). *Pattern classification and scene analysis.* New York: John Wiley & Sons.

Fayyad, U. & Irani, K. (1992). The attribute selection problem in decision tree generation. In *Proceedings of the Tenth National Conference on Artificial Intelligence.* San Jose, Ca. Morgan Kaufmann.

Iba, W. & Langley, P. (1992). Induction of one-level decision trees. In *Proceedings of the Ninth International Workshop on Machine Learning.* Aberdeen, Scotland. Morgan Kaufmann.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence.* Morgan Kaufmann.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability.* Vol. 1. University of California Press, Berkeley.

Murphy, P. and Aha, D. (1994). UCI repository of machine learning databases [machine readable data repository]. Tech. Rep., University of California, Irvine.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA.

Skalak, D. (1994). Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning*, New Brunswick, NJ. Morgan Kaufmann.

Smyth, P. (1996). Clustering using Monte Carlo Cross-Validation. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining.* Seattle, Washington.

Stanfill, C., & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM.* vol. 29. no 12. pp. 1213-1228.

Zhang, J. (1992). Selecting typical instances in instance-based learning. In *Proceedings of the Ninth International Conference on Machine Learning*, New Brunswick, NJ. Morgan Kaufmann.