Symbolic Nearest Mean Classifiers

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

Abstract

The minimum-distance classifier summarizes each class with a prototype and then uses a nearest neighbor approach for classification. Three drawbacks of the minimum-distance classifier are its inability to work with symbolic attributes, weigh attributes, and learn more than a single prototype for each class. The proposed solutions to these problems include defining the mean for symbolic attributes, providing a weighting metric, and learning several possible prototypes for each class. The learning algorithm developed to tackle these problems, 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.

Introduction

The instance-based (Aha, Kibler, & Albert, 1991) or nearest neighbor learning method (Duda & Hart, 1973) is a traditional statistical pattern recognition method for classifying unseen examples. These methods store the training set of examples. To classify an example from the test set, the closest example in the training set is found and the class of that example is predicted. To reduce the number of training examples stored, 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) and others (Niblack, 1986) discuss a variation of the nearest neighbor classifier termed the minimum-distance classifier. This classifier works similarly to the nearest neighbor classifier except that instead of storing each training example, the mean for each class is stored as a prototype. During classification time, the class of closest prototype is predicted. Its main advantage is the significant reduction in storage requirements and classification time. The classification time and storage requirements of nearest neighbor is proportional to the number of examples while the classification time and storage requirements for the minimum-distance classifier is proportional to the number of classes in the domain.

Figure 1a. Figure 1b.

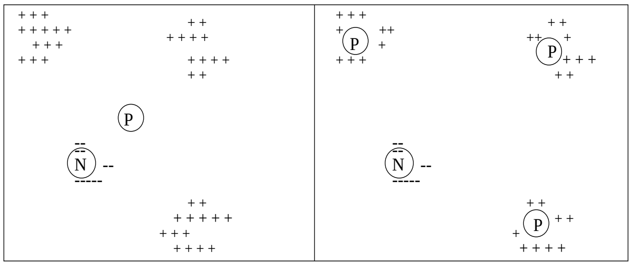

Figure 1: + and − class shown. P and N denote prototypes for the + and − classes respectively. 1a). The single prototype P is not close to the + examples, resulting in misclassification of both + and − examples. 1b). The prototypes resemble the examples better than in Figure 1a, resulting in higher classification accuracy.

Although the minimum-distance classifier has certain advantages, it also has some drawbacks. The minimum-distance classifier 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 that it does not have the ability to weigh attributes. Attribute weighting allows the classifier to decide which attributes are essential for a particular classification task. The last drawback addressed in this paper concerns the inability of the minimum-distance classifier to learn more than one summarization (or prototype) for each class. Using the mean of a class as a prototype will not work well if the prototype is actually distant from the examples as illustrated in Figure 1a. If the minimum-distance classifier could learn multiple prototypes for each grouping of examples (Figure 1b.) then it will be able to better classify in domains where examples are separated into disjoint and distant groups.

In this paper we propose solutions to these problems with the minimum-distance classifier. Section 2 addresses the first problem and 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. SNMC uses clustering to find groupings of the examples for each class. In section 5 we compare these methods with C4.5 (Quinlan, 1993), PEBLS (Cost & Salzberg, 1993), and the minimum-distance classifier using generalization accuracy as our comparison metric. Our empirical results show that SNMC classifies on average better than C4.5 and PEBLS. Section 6 summarizes our contributions and describes directions for future work.

Symbolic Attributes

The distance metric for the 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. This leads to problems in situations where two different values should be considered the same, and in situations where symbolic values should have varying distances among them. In addition, since the minimum distance classifier stores means for each class, a method for computing the mean of symbolic attributes needs to be developed. The most obvious method is to store the most common symbolic value as the mean, however this is incorrect if attribute values can have varying degrees of dissimilarity. Although these two approaches for determining the distance between symbolic values and finding class means are obvious, they have their limitations.

Distances between Symbolic Attribute Values

Stanfill & Waltz (1986) developed the Value Distance Metric (VDM) as a method for computing different distances among symbolic values in a supervised learning task. The VDM has two components, weighing attributes denoted W and finding the distance between two values denoted D. The VDM is not an actual metric since the W component is not symmetric. We only use the D portion of the VDM which is similar to MVDM (Cost & Salzberg, 1991). 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, $f(C_i,S_v)$ denotes the number of examples with class C_i in the set S_v, and $|S_v|$ is the number of examples in S_v. Note that each fraction is between 0 and 1 and thus each difference term will be between 0 and 1. The final distance $D(x,y)$ can be divided by the number of classes C so that the distance between any two values is always between 0 and 1. Since the distance between symbolic attribute values is bounded between 0 and 1, dividing all real values by the range of the real-value attribute (e.g. $\frac{value}{max-min}$) in the training set normalizes them[1]. In domains where both types of attributes are used, the distances for them are then relative and comparable. All of the distances between symbolic attribute values are calculated once and stored in the VDM table before learning takes place.

The Mean of Symbolic Attributes

The simplest method for determining the mean of a symbolic attribute is to choose the most common value. If the distance between symbolic values is constant and equal then choosing the most common value would minimize the variance. However, since we are using the MVDM, the distances between values will not necessarily be equal. In addition, we have the constraint that 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. The class prototypes learned by the minimum-distance classifier can be considered as clusters of examples with the *prototype* described by the mean of the examples in the cluster.

We define the mean of a cluster by finding the value of x_μ that minimizes the variance,

$$Var(J) = \sum_{i \in J}(x_i - x_\mu)^2,$$

where J is a cluster of examples (for minimum-distance classifier, the cluster is the group of examples of the same class) and x_i is a symbolic value for example i. x_μ will be the best constant approximation for the distribution of symbolic values in J in the same way that the mean of real-value attributes is the best constant approximation for the values. Computationally, each symbolic value will be tried as x_μ and the symbolic value that minimizes the variance will become the mean for the symbolic attribute in cluster J.

We call the algorithm that extends the minimum-distance classifier by using the MVDM for symbolic attributes the Symbolic Nearest Mean (SNM). Initially SNM creates the MVDM table and then separates the examples by class. To calculate the mean of the class prototype, the value that minimizes the variance of the examples in a class is found. SNM uses $D(v,w)$

[1] If the range of the attribute in the test set is larger than the maximum value in the training set the test value is set to 1.

for the distance between symbolic attributes and uses $|v - w|$ for real-value attributes. When test examples are classified, the closest class prototype to the test example is calculated using Euclidean distance. The class of the closest prototype is predicted.

Attribute Weighting

Inferring the applicability of an attribute is important for classification. The minimum-distance classifier weighs all of the attributes equally which may decrease classification accuracy. One way of introducing attribute weighting to the minimum-distance classifier is through the MVDM. Since the distances between symbolic values can vary, the MVDM provides a method for weighing symbolic attributes based on classification relevance. For example, if the distances between the values for symbolic attribute A only 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 . Thus the MVDM is indirectly showing A to be less relevant than B for classification.

One advantage of using the MVDM is that it weighs symbolic attributes based on the distributions of their values in the different classes. 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. 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 type 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 even though the actual difference between the number of cigarettes for these two people 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 would want a large difference between persons A and B and a small difference between persons C and D.

Experiments Applying Discretization

To test our hypothesis that discretizing real-value attributes would increase the classification accuracy of SNM, we compared its classification performance with discretization and without. 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 after some initial experiments, we felt that 10 intervals also works with our discretization process. 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. Therefore, the MVDM is more forgiving if slightly too many intervals are created rather than too few.

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

Domain	**SNM**	
	w/o Discr.	**w/ Discr.**
Breast Can.	66.0%	69.9%*
Breast Can. Wis.	95.7%*	91.6%
Credit	84.8%	85.0%
Flag	55.7%	69.5%*
Glass	43.9%	60.2%*
Heart Disease	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%

Table 1 shows the results of 30 runs of this experiment randomly choosing two-thirds of the examples (without replacement) as training examples and the remainder as test examples[2] 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. Both the training and test sets retain the original class probabilities. 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 (an * indicates a statistical significance at the 95% level or higher using a two-tailed t-test) or does not decrease accuracy significantly. In some domains the increase is not great; however, in others it considerably boosts the accuracy of SNM. 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.

[2]We realize that the testing sets in the 30 runs are not independent; however, we apply the t-test as a heuristic to show that the two accuracies can be considered different. Kohavi(1995) recommend multiple runs of cross validation, but we contend that this methodology suffers from the same independence problem.

Learning Multiple Prototypes

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. Using more than one prototype per class allows this method 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. Typically clustering is used in an unsupervised environment for data exploration. Since the examples in an unsupervised environment are not presorted into groups, there is no clear metric to optimize. In our classification task the examples are presorted by class, but we want to find clusters within each of the classes. One way to judge whether a reasonable number of clusters are found in a class is to check the classification accuracy of all training examples. Classification accuracy can be used as a heuristic for determining the number of clusters in individual classes.

The next section describes the method adopted by SNMC (Symbolic Nearest Mean with Clustering), our modification to SNM for representing multiple prototypes within classes. First we describe k-means clustering and then describe a method of deciding on k, the number of clusters for each class.

K-Means Clustering

An area in statistics and machine learning called clustering focuses on the unsupervised task of finding groupings of examples. An efficient algorithm for partitioning a set into k clusters is k-means clustering (Duda & Hart, 1973; Niblack, 1986; MacQueen, 1967). Figure 2 shows the algorithm for k-means clustering. The two main parameters to this algorithm are the distance metric for finding the closest cluster and k, the number of clusters to create. Euclidean distance is the most commonly applied distance measure, although other distance metrics do 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), Schmidt et. al. (1994) and other researchers have suggested various methods to determine the number of clusters in a group of examples; however these are all for learning in an unsupervised environment.

```
Randomly initialize k clusters.
Repeat
    Compute cluster means.
    For each example
        if closest cluster to ex. ≠ current cluster of ex.
            move ex. to closest cluster
Until (no exs. have moved)
```

Figure 2: Pseudocode for k-means.

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

Figure 3: Pseudocode for SNMC.

Finding the Number of Clusters through Training Set Accuracy

Although learning several clusters for each class seems similar to the clustering paradigm, we do have class information available. 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_i is the number of clusters for class i. 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_i when it leads to an increase in accuracy.

Figure 3 shows the procedure for SNMC. SNMC uses a hill-climbing search method. It temporarily increments the number of clusters in each class to see if the classification accuracy on the training set has increased. k_i is only incremented if it results in an increase in classification accuracy on the training set.

SNMC separates the training examples by class and then attempts to find clusters that maximize classification accuracy on the training examples[3]. Afterwards SNMC learns prototypes for each cluster in the same manner as SNM. SNMC predicts classes for unseen test examples by finding the closest prototype and predicting the class of the prototype.

Evaluation of the SNM and SNMC

To evaluate the applicability of these concept descriptions to the classification task, we compared the accuracy of SNM and SNMC with C4.5, PEBLS and

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

the minimum-distance classifier. We compared these algorithms on 20 domains from the UCI Repository. Three of these domains are artificial (LED domains and Hayes Roth) and the remaining are real world domains. These domains have a variety of different characteristics including different types of attributes, and a large range in the number of examples, attributes, and classes.

We describe the methodology for this evaluation. We ran 30 tests randomly picking without replacement two-thirds of the data as the training set and using the remainder as the test set[4] in each of the domains [5] and compared the average accuracy of C4.5, PEBLS, the minimum-distance classifier (Min-dis.), SNM, and SNMC. These results are shown in Table 4. The table represents the accuracy $\pm$ 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 in all of these domains with PEBLS and C4.5 coming in second and third respectively. SNMC has an average increase in accuracy of at least 2% over C4.5 and PEBLS. SNMC also has the best rank of all of the algorithms (2.1); PEBLS and SNM are tied for the next best rank. SNM has a average accuracy slightly lower than C4.5 although not much lower. 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.

When comparing SNMC with C4.5 and PEBLS, we see that SNMC has a higher accuracy than C4.5 in 13 of the 20 domains and SNMC has a higher accuracy than PEBLS in 12 of the 20 domains. When a Wilcoxon test is used to compare SNMC with C4.5 on the 20 domains, where the null hypothesis assumes the accuracy of the two algorithms are equal. This null hypothesis is rejected at the 95% significance level since the statistic between the two algorithms is 110 which is greater than 83.59. The same can be said for comparing SNMC and PEBLS. The computed Wilcoxon statistic is 80 and 77.5 is required for a 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.

The average number of prototypes per class that SNMC learns is fairly small. In the LED domains, where the number of optimal prototypes is known, SNMC learns 1.1 prototypes per class which is close 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.

[4] The training and test sets are stratified.

[5] For the Waveform domain we used 50% of the examples as training examples and the remaining as test examples.

Contributions and Future Work

The first important contribution of this paper is identifying three problems with the minimum-distance classifier, namely its inability to work with symbolic attributes, weigh attributes, and learn disjunctions within classes. This paper describes two algorithms, SNM and SNMC. These algorithms share some core features such as the method they use to define the mean of a cluster, their distance metric, and discretization of real-value attributes. SNMC extends SNM by having the ability to learn multiple prototypes for classes by using k-means clustering. The experimental results show that they perform comparable to both C4.5 and PEBLS in the classification task. In fact SNMC has the highest average accuracy and the best rank in the domains used in our experimentation.

There are several directions for future extensions to this work. Certainly more sophisticated methods for discretizing real-valued attributes can be applied. In addition, there has been a great deal of research on clustering with k-means analysis and they could be applied to SNMC to aid in the clustering process (eg. using the Mahalanobis distance metric). The heuristic used in SNMC could be replaced by a cross-validation method to find the k_i for each class. In addition, the methods described in this paper do not use any prior class distribution information or weigh the prototypes which could boost classification accuracy.

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.

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.

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.

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

Domain	**C4.5**	**PEBLS**	**Min-dis.**	**SNM**	**SNMC**	**proto/cl**
Audiology	88.2±4 (2)	89.4±3 (1)	72.8±6 (5)	82.4±5 (3)	81.9±4 (4)	1.4
Breast Cancer	73.0±3 (1)	67.3±4 (4)	62.0±8 (5)	70.2±5 (3)	72.3±3 (2)	1.3
Br. Can. Wis.	95.1±1 (2)	92.6±2 (4)	95.8±1 (1)	91.8±1 (5)	94.2±2 (3)	1.3
Cl. Heart Dis.	71.9±6 (5)	77.1±3 (4)	80.5±4 (3)	83.5±3 (1)	83.3±3 (2)	1.1
Credit	84.4±2 (3)	82.2±2 (5)	83.1±2 (4)	85.2±2 (2)	85.3±2 (1)	1.1
Flag	72.5±5 (1)	65.3±5 (4)	55.0±6 (5)	67.4±5 (2)	65.7±5 (3)	1.8
Glass	68.4±6 (2)	70.3±5 (1)	41.0±7 (5)	61.1±7 (4)	67.7±6 (3)	1.8
Hayes Roth	81.2±4 (3)	85.5±4 (1)	75.4±5 (4)	38.4±7 (5)	83.2±6 (2)	2.5
Hepatitis	80.3±5 (3)	79.3±6 (4)	77.0±6 (5)	82.9±5 (1)	82.5±5 (2)	1.5
House Votes	96.1±1 (1)	94.2±1 (3)	87.3±3 (5)	90.4±2 (4)	94.9±2 (2)	2.9
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
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
LED w/irr. feat.	67.3±4 (3)	62.8±4 (4)	29.7±4 (5)	73.0±3 (1)	72.5±3 (2)	1.1
Lymphography	79.5±5 (4)	82.9±4 (1)	71.0±7 (5)	82.1±4 (3)	82.5±5 (2)	1.4
Pima Ind. Dia.	72.1±3 (4)	69.4±3 (5)	73.1±2 (3)	75.3±3 (2)	75.8±3 (1)	1.4
Promoters	78.3±7 (4)	89.0±4 (3)	71.3±9 (5)	91.4±5 (1)	90.6±5 (2)	1.8
Soybean-large	86.8±4 (3)	92.4±2 (1)	68.4±5 (5)	82.2±4 (4)	86.9±4 (2)	1.3
Waveform	74.9±.9 (5)	77.7±.9 (4)	79.2±6 (3)	81.1±.8 (2)	81.4±1 (1)	1.0
Wine	92.7±4 (5)	97.4±2 (1)	95.6±2 (4)	96.1±3 (3)	97.1±2 (2)	1.3
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

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.

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.

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.

Niblack, W. (1986). *An Introduction to Digital Image Processing.* Prentice Hall.

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

Schmidt, W. Levelt, D. & Duin, R. (1994). An experimental comparison of neural classifiers with 'traditional' classifiers. Pattern Recognition in Practice IV: Multiple Paradigms, Comparative Studies, and Hybrid Systems, Proceedings of an International Workshop. Vlieland, The Netherlands, Elsevier.

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 em Proceedings of the Ninth International Conference on Machine Learning, New Brunswick, NJ. Morgan Kaufmann.