Cybernetics and Systems: An International Journal, 23:483–515, 1992

RCE CLASSIFIERS: THEORY AND PRACTICE

MICHAEL J. HUDAK

38 Oliver Street, Binghamton, New York 13904

Restricted Coulomb Energy (RCE) classifiers, as described by Scofield et al. (1988), are shown to have a conceptual relationship with hyperspherical classifiers developed in the 1960s by Batchelor (1974). These classifiers are also shown to share similarities with networks of localized receptive fields and with psychological models of concept formation. Next, the performance of some RCE classifiers is examined. The ability of a trained RCE classifier to generalize to new instances is compared with that of several well-known classifiers. Then, four previously unexamined aspects of RCE classifiers are investigated empirically: (1) the influence of potential wells on training rate, (2) the influence of potential wells on storage requirement, (3) the influence of potential wells on generalization to new instances, and (4) rejection of an instance from an unknown class. Modifications of a traditional RCE classifier improve average correct classification at generalization from 83.2 to 90.7% without significant change in computational cost. By comparison, a nearest-neighbor performs at 93% and a feed-forward multilayer neural network at 88.4% on the same data. Surprisingly, when the improved RCE network is compared with its underlying adaptive nearest-neighbor component, one finds that the incorporation of potential wells into the RCE classifier does not reduce training time or instance storage requirement, nor does it improve generalization to new instances.

INTRODUCTION

Pattern classifiers constructed from the union of hyperspherical decision regions have been studied since the early 1960s. The earliest work (e.g., Cooper, 1962, 1966) was primarily theoretical in nature. Later empirical work by Batchelor (1974) compared the shape of decision regions for hyperspherical classifiers with those of other nonparametric classifiers. Not until the work of Lee (1989) and Lee and Lippmann (1990) was there a thorough empirical examination of hyperspherical classifiers and their special case, Restricted Coulomb Energy (RCE) classifiers.

Hyperspherical classifiers have attracted attention for several reasons and have been viewed in several ways, as demonstrated by the following list.

Although resembling the nearest-neighbor classifier (Duda and Hart, 1973) in its storage of prototypical patterns, a hyperspherical classifier is typically more conservative of storage (Batchelor, 1974: 49) and can abstain from classifying a pattern from an unknown category (Li, 1988: 21).

Although a hyperspherical classifier is typically trained by repeated presentation of training instances, as is required in backpropagation training of feedforward neural networks, hyperspherical classifiers typically require considerably fewer training epochs (complete presentations of training instances) than do neural networks (Lee, 1989; Lee and Lippmann, 1990).

A hyperspherical classifier has been viewed as an "exemplar-based" model of biological memory in which memory is constructed from representation of experiences (Kanerva, 1988).

Suitability of implementing hyperspherical classifiers in hardware has been recognized in several works (Flynn et al., 1988; Potter, 1987; Reilly et al., 1987; Rimey et al., 1986).

A form of hyperspherical classifier has been conceptualized as a neural network and termed the RCE model. This provides a high-density memory system unencumbered by the storage limitations (Hopfield, 1982) of single-layer, recurrent networks (Bachmann et al., 1987; Scofield et al., 1988).

Hyperspherical receptive fields have been used as a front end to a trainable neural network classifier (Moody and Darken, 1989, Nowlan, 1990).

This study has three major purposes: (1) to show the relationship between the RCE classifier (Scofield et al., 1988) and the hyperspherical classifier as described by Batchelor (1974); (2) to point out similarities of hyperspherical classifiers to some work in neural networks and cognitive psychology; and (3), building on the experimental work of Li (1988) and Lee (1989), to explore several practical issues that arise when RCE classifiers are applied to classification problems with real data.

HISTORICAL OVERVIEW

Batchelor's Compound Classifier

Types of hyperspherical classifiers were introduced by Cooper (1962, 1966) and Batchelor (1968, 1969); a summary is provided in Batchelor (1974). Like the nearest-neighbor classifier, the hyperspherical classifier is based on the storage of instances (each represented as a numeric vector called a "pattern") from known classes creating points in a metric space. The distance function defined on this space provides the basis for assigning a pattern of unknown class to a known category. Unlike the nearest-neighbor classifier, each point [also called a "locate" by Batchelor (1974)] is associated with a radius that defines the point's region of influence. In other words, each locate-radius pair defines a decision region associated with the class of the locate.

Batchelor (1974) conceived of the hyperspherical compound classifier as a layered device:

The bottom layer contains subclassifiers. Each subclassifier decides whether an input instance lies inside a hypersphere.

The top layer combines the subdecisions from the bottom layer by forming their logical union.

The formulation for the 2-class case (i.e., for a classifier capable of distinguishing two classes) is provided as an example. Assume there are N_1 stored patterns in class 1: $\vec{p}_{11}, \ldots, \vec{p}_{1N_1}$. The radii of the stored patterns are represented by $R_{11}, \ldots, R_{1N_1}$. the ith subdecision U_i is calculated as

$$U_i = sign\,(R_{1i} - D_{1i}) = \begin{cases} 1, & \text{if } \| \vec{p}_{1i} - \vec{x} \|_2 < R_{1i} \\ 0, & \text{if } \| \vec{p}_{1i} - \vec{x} \|_2 = R_{1i} \\ -1, & \text{if } \| \vec{p}_{1i} - \vec{x} \|_2 > R_{1i} \end{cases} \tag{1}$$

where D_{1i} is the Euclidean distance defined by

$$D_{ij} = \Big(\sum_k (p_{ijk} - x_k)^2\Big)^{\frac{1}{2}} \tag{2}$$

and $\vec{x}$, the instance to be classified, is represented by coordinate values x_1, . . . , x_k.

The complete decision of the classifier (i.e., the class to which the instance $\vec{x}$ is supposed to belong) is then calculated as

$$C = \begin{cases} 1, \text{ if } \bigcup_{i=1}^{N_i} (U_i \equiv 1) \\ 2, \text{ otherwise} \end{cases} \tag{3}$$

Batchelor (1974) developed procedures that construct a hyperspherical compound classifier from a set of instances of known class. These procedures require the iterative presentation of instances until either they are all correctly classified, an equilibrium is established (i.e., further changes of radii and locates are insignificant), or a limit on iterations is exceeded. The elements of his approach are given below for the 2-class case. In following Batchelor's notation, *T*(*teacher*) is the actual class to which the training instance $\vec{x}$ belongs; *C*(*classifier*) is the class of $\vec{x}$ according to the previous decision of the classifier. One of three situations arise upon each presentation of a training instance:

1. C and T are equal. No changes are necessary.
2. $C = 2$, $T = 1$ ($\vec{x}$ lies outside all hyperspheres). Find the hypersphere closest to $\vec{x}$. Then move the locate of this hypersphere toward $\vec{x}$ and increase the radius of the hypersphere.
3. $C = 1$, $T = 2$ ($\vec{x}$ lies inside any of the hyperspheres). The locate of each hypersphere whose class is 1 that encloses $\vec{x}$ is moved away from $\vec{x}$, and its radius is reduced.

Batchelor's training algorithms based on these principles require incremental adjustment of two parameters in the classifier:

1. Locates (which begin as instances from the training set).
2. Radii (each of which is associated with a locate).

Determination of the step sizes for each of these increments is troublesome. Designating the incrementation parameter for locate moving as k and for radius adjustment as l, Batchelor (1974: 132–133) stated:

> At the end of a long learning sequence, the classifier is in a state of dynamic equilibrium, with a small tremor on its locates and weights. The magnitude of this vibration increases with *k* and *l*. When we stop learning, we fix the classifier at some arbitrary point in this vibrating equilibrium state. The accuracy of this fixed classifier is likely to be reduced if *k* and *l* are large. On the other hand, the learning will taken an unduly long time if very small values of *k* and *l* are used. . . . Certainly, we should never use larger values, although if computer time is available, smaller values can be used. It should be possible to reduce *k* and *l* progressively, in such a way that we obtain rapid learning, with a precise classifier available afterwards. We do not yet know how this should be done to achieve optimal results.

Batchelor (1974: 130–134) extended this 2-class definition to an arbitrary number of classes with an *N*-class classifier consisting of *N* 2-class compound classifiers. The *i*th subclassifier is individually trained to separate the *i*th class from all others. During testing the classifiers are operated in parallel. Ambiguity in the classification is detected by recognizing the simultaneous outputs from two or more subclassifiers.

Batchelor (1974: 133) remarked that for the ranges of *k* and *l* he had tested that he "never encountered any significant changes occurring after about 5000 iterations." (An "iteration" refers to one presentation of all training patterns, i.e., a training epoch.) Nevertheless, Batchelor (1974) suggested allowing 10,000 iterations before invoking his procedures to add more locates (growing), or removing redundant ones (pruning).

Summary of Batchelor's Work

Details of Batchelor's procedures for training, growing, and pruning his compound classifier have been omitted here; the interested reader is referred to Batchelor (1974) for more information. The most significant characteristics of his basic classifier are summarized below.

Lack of concern for stored instances (points) as "prototypes": Stored points can be moved in an attempt to minimize storage of instances from the training set. Hence, a stored point (called a locate by Batchelor) typically becomes unlike any instance in the training set and only defines the center of a hypersphere.

Incremental training: A locate and its radius are incrementally modified to produce a decision region. Every locate in the classifier is a candidate for modification. Typically, up to 5000 training epochs are needed to classify all training instances correctly.

Concern for minimizing the number of locates in the classifier: New training instances are not stored unless the classification error over the training set exceeds a user-specified threshold and further improvement appears unlikely from modification of locates and radii. Furthermore, redundant or little-used locates can be removed by pruning.

No suggestion for disambiguation of classes: In the 2-class classifier formulated by Batchelor the region within hyperspheres is one class and the region outside any hypersphere is another class. Ambiguity of class membership cannot exist. For more than two classes, Batchelor proposed a method that detects ambiguity but cannot resolve it.

RCE Networks

RCE classifiers are inspired by systems of charged particles in a three-dimensional space. I intend to show the relationship between this model, in its implementation (Scofield et al., 1988), and Batchelor's compound classifier (Batchelor, 1974). The following brief presentation leading up to the RCE model uses the notation of Bachmann et al. (1987) and Scofield et al. (1988) for the sole purpose of preventing a conflict for the reader who directly consults these works.

The classifier's name (RCE) pertains to the form of the system's Liapunov function (equation 4), which is isomorphic to the classical electrostatic potential between a positive test charge and negative charges Q_i at the sites $\vec{x}_i$ (for three-dimensional input space and $L = 1$). Viewed another way, the $\vec{x}_i$ form a set of distinct memories in R^N.

$$\xi = -\frac{1}{L}\sum_{i=1}^{m} Q_i \| \vec{\mu} - \vec{x}_i \|_2^{-L} \tag{4}$$

where ξ is the electrostatic potential induced by fixed particles with charges $-Q_i$, $\vec{\mu}$ is a vector describing the network state, $\vec{x}_i$ is the site of the ith memory for m memories in R^N, and L is a parameter related to the network size. Then $\vec{\mu}(t = 0)$ relaxes to $\vec{\mu}(t = T) = \vec{x}_i$ for some memory i according to

$$\dot{\vec{\mu}} = \vec{E}_{\vec{\mu}} \triangleq -\sum_{i=1}^{m} Q_i \| \vec{\mu} - \vec{x}_i \|_2^{-(L+2)} (\vec{\mu} - \vec{x}_i) \tag{5}$$

which "describes the motion of a positive test particle in the electrostatic field $\vec{E}_{\vec{\mu}}$ generated by the negative fixed charges $-Q_1, \ldots, -Q_m$ at $\vec{x}_1, \ldots, \vec{x}_m$" (Bachmann et al., 1987: 7530). Describing this system, Scofield et al. (1988: 676) stated:

> The N-dimensional Coulomb energy function then defines exactly m basins of attraction to the fixed points located at the charge sites $\vec{x}_i$. It can be shown (Bachmann et al., 1987; Dembo and Zeitouni, 1988) that convergence to the closest distinct memory is guaranteed, *independent of the number of stored memories m,* for proper choice of N and L. Equation (4) shows that each cell receives feedback from the network in the form of a scalar
>
> $$\sum_{i=1}^{m} Q_i \| \vec{\mu} - \vec{x}_i \|_2^{-L}. \tag{6}$$
>
> Importantly, this quantity is the same for all cells; it is as if a single virtual cell was computing the distance in activity space between the current state and stored states. The result of the computation is then broadcast to all of the cells in the network.

Referring to an equilibrium feedforward network with similar properties described in Reilly et al. (1982), Scofield et al. (1988: 676–677) continued:

> This model does not employ a relaxation procedure, and thus was not originally framed in the language of Liapunov functions. However, it is possible to define a similar system if we identify the locations of the "prototypes" of this model as the locations in state space of potentials which satisfy the following conditions
>
> $$\xi_i = \begin{cases} -Q_i / R_o, & \text{if } \| \vec{\mu} - \vec{x}_i \|_2 < \lambda_i \\ 0, & \text{if } \| \vec{\mu} - \vec{x}_i \|_2 > \lambda_i \end{cases} \tag{7}$$

where R_0 is a constant.

This form of potential is often referred to as the "square-well" potential. This potential may be viewed as a limit of the N-dimensional Coulomb potential, in which the $1/R(L = 1)$ well is replaced with a square well (for which $L \gg 1$). Equation (7) describes an energy landscape which consists of plateaus of zero potential outside of wells with flat, zero slope basins. Since the landscape has only flat regions separated by discontinuous boundaries, the state of the network is always at equilibrium, and relaxation does not occur. For this reason, this system has been called an equilibrium model. This model, also referred to as the Restricted Coulomb Energy (RCE) model, shares the property of unrestricted storage density.

The relationship between the RCE network classifier and Batchelor's compound classifier is straightforward. Both systems store training instances, and Batchelor's hypersphere radii (R_{ij}) correspond to the RCE classifier's potential well boundaries (λ_i) [Eq. (7)]. Significant differences in the formulations arise from the RCE classifier's generalization to multiple classes, the meaning of a "locate" or "memory," and the treatment of hypersphere radii and potential well boundaries. Although Batchelor requires that all stored patterns belong to one class in a given instance space, the RCE classifier provides for multiple classes in the instance space simply by storing instances from multiple classes. Furthermore, unlike Batchelor's formulation, memories in an RCE classifier cannot be modified: once an instance is stored it can be removed but not moved to another location. Finally, although Batchelor's hypersphere radii can both increase and decrease, an RCE potential well boundary can only decrease as training proceeds.

A simple training procedure for the RCE classifier is provided in Reilly et al. (1982) and is consistent with the characterization presented in Lee (1989).

Networks of Localized Receptive Fields (NLRF)

Hyperspherical classifiers have a close relative in networks of localized receptive fields (NLRF) (Moody and Darken, 1989), also known as radial basis functions (Nowland, 1990). Although evidence does not exist of interaction between research on hyperspherical classifiers and NLRF, examining their similarities and differences will provide additional perspective on the hypersphere paradigm. The work of Moody and Darken (1989) will be cited as representative of its genre.

NLRF classifiers were developed in response to extremely long training times encountered with backpropagation trained feedforward networks. In this sense the reduced training requirement of NLRF classifiers, relative to feedforward networks, parallels that of RCE networks to Batchelor's hyperspherical classifiers. Noteworthy comparisons between hyperspherical classifiers and NLRF classifiers are sixfold.

1. Terminology: "Center of receptive field" in NLRF literature is synonymous with the terms "hypersphere center," "locate," and "stored point" encountered in writings about hyperspherical classifiers, but methods for determining receptive field centers and locates are not identical.
2. Distance of influence for a "center": The influence of a hypersphere center in a hyperspherical classifier is limited by the radius of its hypersphere. In an NLRF classifier the influence of a center extends to infinity but with an intensity that diminishes exponentially with the distance from the center.
3. Selection of number of centers: Moody and Darken (1989) proposed methods for NLRF classifiers that require the a priori selection of the number of receptive field centers for the set of training instances. Typical training algorithms for hyperspherical classifiers do not have this restriction.
4. Location of centers: Moody and Darken (1989) recommended forming centers from the set of training instances with either standard k-means clustering or its adaptive formulation. As is typical of locate formation in Batchelor (1974), cluster centers usually do not correspond to any training instance.
5. Creation of receptive fields: Moody and Darken (1989) proposed determining the "width" of each receptive field after all the centers are chosen. Width is similar to the concept of standard deviation of a distribution. Hypersphere radius is also similar to this width, but the radius puts a hard limit on the center's influence. Furthermore, training algorithms for hyperspherical classifiers do not separate, in this manner, the processes of hypersphere center selection and radius computation.
6. Combining information from receptive fields: NLRF classifiers as described by Moody and Darken (1989) employed output weights called

receptive field amplitudes that are adjusted by a least mean square learning rule. These weights combine the responses of the receptive field functions in producing a single classification. In contrast, Batchelor's (1974) compound classifier does not contain weights. An unknown instance falling within a hypersphere is assigned the class associated with that hypersphere. If an instance falls within hyperspheres belonging to more than one class, the classifier is unable to resolve the ambiguity. RCE classifiers (Scofield et al., 1988) allow for associating a potential well "depth" with each stored point; depth is similar to the concept of weight in an NLRF.

Models from Cognitive Psychology

Two models of concept representation in cognitive psychology show strong parallels to the compound classifier of Batchelor (1974) and the RCE classifier of Scofield et al. (1988). Much of this overview is taken from Matheus (1987), who outlined the models termed the "prototype view" and the "examplar view."

The prototype view of concept formation posits that a concept is represented by a summary description in which the features need merely be characteristic of the concept instances. "The summary description is called the 'prototype' of the concept and encodes the most central features of the concept's instances. A prototype is an abstract representation and need not depict an actual instance of the concept" (Matheus, 1987: 45). Notice the similarity between prototypes and Batchelor's locates. Although a locate begins as a training instance, its ability to "drift" during training can lead to a collection of feature values unlike those possessed by any instance of the class. Note that the features in a hyperspherical classifier must be numeric whereas those of the prototype view need not be. Although the prototype model (e.g., Posner and Keele, 1970) dates from the same period as Batchelor's (1968, 1969) early work, evidence does not exist of cross-fertilization between these two fields.

Matheus (1987) pointed out major strengths and weaknesses of the prototype model. Given its relationship to Batchelor's compound classifier, these characteristics should be considered if this classifier is incorporated into a larger learning system. In favor of the prototype model, Matheus noted that it mimics human response in unclear and in typical cases. On the other hand, most prototype representations retain no information about each feature's range of values. Only information about the typical value is retained. Consequently, correlational information among a concept's features is also lost.

As an alternative to the prototype model, Matheus (1987) presented what he called the exemplar view. Instead of constructing an explicit summary description for each concept, an exemplar model represents a concept as a collection of instances or exemplars (Medin and Schaffer, 1978; Medin and Smith, 1984). In a pure exemplar model (Reed, 1972) all exemplars exist as descriptions of individual instances. The relationship between the exemplar model and the RCE classifier is straightforward. Both systems store instances as collections of features that remain unchanged as learning proceeds, but, unlike the RCE classifier, these exemplar models do not surround exemplars with hyperspheres.

In contrast to the prototype model, the exemplar model provides for simpler concept formation and modification. Because a concept is represented by a collection of exemplars, modification occurs each time a new instance is added to the concept's description—no information about a feature's values is lost as new exemplars are encountered. Hence the range of encountered feature values within a concept is retained, allowing the examination of feature correlation. The major defect of the exemplar approach is its enormous storage requirement for complex concepts. Although Matheus (1987) suggested that storage can be reduced by the selective collapse of some exemplars into prototypes, he did not explain how this can be done.

Experimental Investigations

The performance issues of RCE classifiers addressed in this paper rest on the experimental work of Li (1988) and Lee (1989). A brief summary of their work will provide an understanding of the questions they answered and the ones they did not explore.

Li (1988) examined the feasibility of a hypersphere-based classification system for doing low-level diagnosis of computer modules. She tested six training algorithms, elements of which are derived from Batchelor (1974). Two algorithms use incremental modification of radii; the remainder perform large-scale radius reductions, each reduction being sufficient to exclude at least one locate from a conflicting class. Half the algorithms allow locates to move, as described in Batchelor (1974); the remainder require locates to remain stationary, as in the RCE classifier of Scofield et al. (1988).

A database of 124 module signatures drawn from 35 different classes was divided into a training set of 76 patterns and a test set of 48 patterns.

Among the six algorithms, differences in the storage required are reportedly insignificant. Test performance ranged from 54 to 62% classification accuracy, defined as the predicted electronic component being either the first item on the prediction list or one of the ties for the first position. Best performance was achieved by an algorithm using incremental modification of radii and locates. Since these algorithms are sensitive to arbitrary parameters and the ordering of the training data, it is not possible to determine whether these factors, or the invariant aspects of the algorithms, account for the differences in performance. Multiple trials would have provided some insight into this matter.

Lee (1989) and Lee and Lippmann (1990) provided an evaluation of eight classifiers, including a version of Batchelor's hyperspherical compound classifier and its special case, the RCE classifier. In evaluating the hyperspherical classifier, Lee (1989: 118–121) reported the three significant points summarized below:

1. Decreasing the value by which hyperspheres were incrementally modified greatly prolonged training and improved generalization accuracy by only 1 to 2%.
2. When the number of hyperspheres was not limited by memory constraints, the improvement in generalization accuracy resulting from moving locates was insignificant. When memory was limited, moving locates significantly improved generalization accuracy, but the level achieved did not exceed that obtained with unconstrained storage.
3. Two criteria were examined for removing a locate from the classifier: the size of its hypersphere and the locate's effectiveness in correctly classifying training instances. Although a locate's effectiveness proved the better indicator of its contribution to test set classification, Lee did not report the degree of improvement.

Lee then proceeded with a closer look at the RCE classifier, which he characterized as a special case of "a generalized version of algorithms from the work of Batchelor (1974, 1978)" (Lee, 1989: 72). Lee's experiments, conducted with four databases and eight classifiers, yielded several major results about the RCE classifier. Of primary concern are the following:

The RCE classifier's generalization error exceeded the weighted average error of all classifiers by approximately 1 standard deviation on three of the databases and equaled the weighted average error in the remaining case (Lee, 1989: 133–134).

The RCE classifier trained, on the average, in 11% of the weighted average time of all the classifiers (Lee, 1989: 131).

Like the nearest-neighbor classifier, the RCE classifier stores instances from a training set. But whereas the nearest-neighbor classifier stores the entire set, the RCE classifier typically stores a minority of training set instances. Storage requirements for the RCE classifier ranged from 5.4 to 29.5% of the training instances, with the weighted average of 12.1% (Lee, 1989: 130). Values are for an "unpruned" classifier—one in which little-used instances have not been removed.

To summarize, although Lee's RCE classifier demonstrated favorable training times and memory requirements, it proved deficient in the correct classification of new instances compared with the other classifiers.

EXPERIMENTS

Experimental Background

Although Li (1988) and Lee (1989) performed extensive empirical studies of many hyperspherical classifiers, including RCE, several basic questions remained unanswered. Four such issues addressed in my study appear below.

1. Although Lee (1989) compared an RCE classifier with seven well-known classifiers, the experiment that would have tested the significance of hyperspheres was not performed. That experiment compares an RCE classifier to its underlying incremental nearest-neighbor component.
2. Storage requirement, training epochs, number of hyperspheres modified, and generalization to new instances are all affected by the order in which training instances are presented to a hyperspherical classifier. For Batchelor (1974), instance ordering was crucial in avoiding oscillation of locates and nonconvergence of a global error function over the training instances. Elimination of hypersphere enlargement in the RCE classifier avoids these problems but brings the issues mentioned above to prominence.
3. Lee (1989: 133–134) showed that the hyperspherical classifier is deficient in its ability to generalize, compared with the other classifiers he tested. Nevertheless, he did not propose a method for improving this performance.

4. Li (1988: 21) stated that the hyperspherical "classifier can return *no match* when the input pattern does not fall into any hypersphere in the *n*-space." That a hyperspherical classifier can, in theory, report that a test instance does not belong to a known category is obvious; yet this feature has not been tested for any type of hyperspherical classifier.

Data Preprocessing

Implementation constraints for the various RCE classifiers, the nearest-neighbor, and the incremental nearest-neighbor (INN) classifier require that each feature value be mapped to one of the 256 decimal integer values (0–255). Maximum and minimum values for each feature of the training patterns were mapped to the interval end points, with intermediate values appropriately scaled. The range between end points was then used to scale the test patterns. The remaining classifiers—cascade-correlation, quickpropagation trained multilayer feedforward neural network, and decision tree classifier—do not require preprocessing of instances.

Databases

Experiments were performed with three public domain databases: the Iris database of Fisher (1936), the Sonar database of Gorman and Sejnowski (1988), and the Congressional voting records database of 16 key votes from the 98th Congress, 2nd session, 1984. Database characteristics and details of the instance sets used for training and generalization are provided in Appendix A.

Experimental Regime

Unless stated otherwise, the values reported for the RCE classifiers and INN are based on 100 training trials. Specifically, although trials were conducted with the same sets of training and test instances, each trial was performed with a unique random ordering of the training instances. From these trials, the average value and the standard deviation of the average are reported. The effect of training instance order on performance is thus revealed. Furthermore, expected levels of performance are established for each classifier that can then be used in meaningful comparisons.

All the RCE classifiers employ a user-specified parameter (DIST_PCT) that reduces a hypersphere radius during training to a percentage of the distance between hypersphere centroids from different classes. A complete description of this parameter and the classification algorithms written in an ALGOL-like pseudocode can be found in Appendix B. Trials were performed with DIST_PCT set at each of the values 15, 35, 55, 75, 95, and 99%. Except where noted, the results presented were obtained with DIST_PCT set at 99%, which consistently yielded the highest level of generalization while correctly classifying the entire set of training instances.

All statistical tests performed on the experimental results rest upon two basic assumptions:

1. That the sample size is sufficiently large for the sample parameters to provide a good approximation to those of the underlying population.
2. That the values are normally distributed.

In accordance with these assumptions, z scores (Dowdy and Wearden, 1991: 207–208; Sachs, 1982: 272) are used in all hypothesis tests.

Throughout the presentation of results the hypothesis is tested that a pair of performance means from different classifiers are the same. In other words, does a difference in classifier design result in different performance? In several cases the performance values are expressed as percentages. One reviewer has pointed out that distributions of percentages near the ends of the range (<20%; >80%) tend to be binomial rather than normal. The remedy is to transform each distribution to better approximate a normal distribution before performing tests of hypotheses about the means (Alder and Roessler, 1972). The transformation used here is the arcsin of the square root of each percentage yielding a value in degrees. Means and standard deviations are computed from these values before computing z scores used in the hypothesis test.

In the tables, classifiers are ranked from best (top) to worst (bottom) with respect to weighted average performance. Weighted averages are computed from the performance on each database and the percentage of either training or testing instances in the database, relative to the total number of such instances from all databases.

Classifier Comparisons

Although this study is primarily about the RCE classifier, its performance compared with that of other classifiers is important in establishing is relative worth. For this purpose, the four classifiers listed below are also tested:

Feedforward multilayer neural network with quickpropagation training (Fahlman, 1988)
Cascade-correlation neural network (Fahlman and Lebiere, 1990)
Decision tree classifier (c4.5) (Quinlan, 1986)
Nearest-neighbor classifier (Duda and Hart, 1973).

Computer code for the first three classifiers listed above was obtained from the authors. A basic RCE classifier known as the disjoint spheres/no drift (DSND) algorithm (Li, 1988: 62) is used for comparison.

Table 1 displays the ability of DSND to generalize, together with that of the other four classifiers. Training of the quickpropagation and cascade-correlation networks proceeded until all training instances were correctly classified for each of 10 randomly selected initializations of network weights. These two classifiers' strong and unpredictable dependence on initial weight values is surprising. One plausible explanation for this dependence is the use of asymmetric error correction to each training instance's target value (R. L. Watrous, personal communication).

Note that the methodology for training and testing quickpropagation on the Sonar database differs from that employed by Gorman and Sejnowski (1988). Consequently, these results are not comparable with theirs.

From Table 1 the following conclusions can be drawn. First, that an RCE classifier can successfully compete with several well-known classifiers

TABLE 1. Correct Classification of New Instances

	Instances classified (%)			
Classifier	Iris database	Sonar database	Voting database	Weighted average
Nearest-neighbor ($k_g = 1$)	96.0	95.2	91.2	93.0
Quickpropagation	91.1 ± 2.1	75.5 ± 2.6	93.9 ± 1.5	88.4
Cascade-correlation	94.2 ± 2.1	74.9 ± 3.1	91.4 ± 0.0	87.1
c4.5	96.0	65.4	94.5	86.5
DSND	89.8 ± 3.1	83.5 ± 2.5	81.0 ± 3.0	83.2

Averages and standard deviations for quickpropagation and cascade-correlation networks obtained from 10 different initializations of network weights. Number of hidden units chosen for the quickpropagation trials: Iris database, 2 or 3; Sonar database, 4; Voting database, 2. Single trials are reported for c4.5 and nearest-neighbor classifier, because performance is unaffected by training instance order.

in its ability to generalize, but that its performance tends toward the lower end of the range; this corroborates the finding of Lee (1989). Second, that if storage is not limited, the nearest-neighbor classifier is the best overall choice; this finding is, for the most part, also consistent with Lee (1989).

Significance of Hyperspheres

Although Cooper (1962, 1966) showed the optimality of hyperspherical decision boundaries for particular probability distributions, the significance of the hypersphere has not been demonstrated in classifiers of the types described by Batchelor (1974) and Reilly et al. (1987). Although RCE classifiers have been compared with other well-known classifiers (e.g., the previous section, Lee, 1989), each of the two basic components of the RCE classifier has not been examined for its individual contribution to the classification process. These components are (1) locates (represented as numeric vectors) and an associated distance metric and (2) a hypersphere (potential well in RCE terminology) about each locate that limits the attraction of the locate to a test instance. My approach for examining the contribution of hyperspheres to the classification process was to construct two RCE training algorithms (RCE-1 derived from DSND and RCE-2 derived from RCE-1), each of which successively limits the influence of its hyperspheres. Finally, a third classifier INN, derived from RCE-2, in effect maintains hyperspheres with infinite radii, hence eliminating the influence of hypersphere boundaries in the classification process. INN ($k_t = k_g = 1$) is essentially the IB2 algorithm of Aha et al. (1991), differing only in the choice of distance metric. Here k_t is the number of nearest neighbors used in training and k_g is the number used in generalization. INN can be regarded as the nearest-neighbor component underlying the RCE classifiers examined in this study. Details of all the RCE classifiers and INN may be found in Appendix B.

The role of hyperspheres is of interest in at least three aspects of the classification process:

1. Generalization to new patterns from known classes.
2. Percentage of the training set instances needed to classify all training instances correctly.
3. Number of training epochs needed to classify all training instances correctly.

TABLE 2. Correct Classification of New Instances

	Instances classified (%)			
Classifier	Iris database	Sonar database	Voting database	Weighted average
INN ($k_t = k_g = 1$)	96.6 ± 2.0	91.9 ± 2.4	89.5 ± 1.8	91.1
RCE-2	95.0 ± 2.4	91.8 ± 2.2	89.2 ± 2.0	90.7
RCE-1	91.7 ± 3.2	85.3 ± 2.4	85.7 ± 2.4	86.4
DSND	89.8 ± 3.1	84.5 ± 2.5	81.0 ± 3.0	83.2

We begin with the essential factor, generalization. The results are shown in Table 2. The classifier performance shown in Table 2 was subjected to the test of the hypothesis that selected pairs of classifiers produce equal means. As previously described, in regard to percentages, the hypothesis testing was conducted on transformed data underlying the values shown in Table 2. For the Iris database the hypothesis is rejected at the 0.0002 level for all distinct pairs of classifiers. For the Sonar and Voting databases the hypothesis is rejected (0.0002 level) for all distinct pairs except (INN, RCE-2). These results strongly indicate that the algorithm differences result in significant performance differences by these classifiers.

Next, the effect of hyperspheres on the number of training epochs needed to learn the set of training instances is displayed in Table 3. As in Table 2, the RCE classifiers only approach the incremental nearest-neighbor classifier INN in performance.

Lastly, turning to the issue of classifier size, the percentage of instances from the set of training instances are needed to classify all training instances correctly. The results, shown in Table 4, indicate that only in the case of the Iris database is RCE-1 significantly more efficient of memory than is INN ($k_t = k_g = 1$). Specifically, in a two-tailed test, for the Iris database the

TABLE 3. Number of Training Epochs Needed to Classify All Training Instances Correctly

Classifier	Iris database	Sonar database	Voting database	Weighted average
INN ($k_t = k_g = 1$)	2.3 ± 0.6	3.2 ± 0.8	2.5 ± 0.6	2.6
RCE-2	2.4 ± 0.7	3.2 ± 0.7	2.9 ± 0.7	2.9
DSND	2.6 ± 0.7	3.7 ± 0.7	3.3 ± 0.7	3.3
RCE-1	3.0 ± 0.9	4.2 ± 1.0	3.2 ± 0.8	3.4

TABLE 4. Percentage of the Training Set Instances Required to Classify All Training Instances Correctly

	Instances required (%)			
Classifier	Iris database	Sonar database	Voting database	Weighted average
INN ($k_t = k_g = 1$)	17.5 ± 2.0	49.1 ± 3.5	12.7 ± 1.9	23.5
RCE-1	13.5 ± 2.7	47.6 ± 3.6	14.5 ± 1.9	23.6
RCE-2	17.4 ± 2.3	56.0 ± 3.3	15.9 ± 1.7	27.3
DSND	17.3 ± 1.8	60.8 ± 2.3	21.4 ± 2.0	31.9

hypothesis that INN ($k_t = k_g = 1$) and RCE-1 produce the same mean is rejected at the 0.0002 significance level. Note the ineffectiveness of hyperspheres in reducing storage requirements when the training instances exhibit highly nonlinear class separation as in the Sonar database. Here INN is markedly superior in storage efficiency to RCE-2.

To summarize, a few simple modifications to the original RCE classifier DSND, yielding the classifier RCE-2, improve correct classification of new instances from 83.2 to 90.7%. Nevertheless, viewing RCE-2 as an incremental nearest-neighbor classifier with hyperspheres leads to the conclusion that hyperspheres are not positively contributing to the performance of RCE-2. By all measures, management of hyperspheres is an unjustified computational cost.

Proposal for Improved Generalization

Nearest-neighbor classification better approximates the Bayes error rate as the number of samples k used in classification is increased, provided that k is a small fraction of the stored instances (Duda and Hart, 1973: 105). Known as the k-nearest-neighbor rule, k is typically chosen as 3, or 5 for the 2-class case so that one class will always have a majority of votes. When the number of stored samples is small, though, it is not necessarily beneficial to take k large because nonlocal samples tend to influence the decision.

Viewing RCE-2 as a specialization of INN ($k_t = k_g = 1$) suggests that the generalization of nearest neighbor ($k_g = 1$) to ($k_g > 1$) can be applied to RCE classifiers. Whereas in the nearest-neighbor classifier a large number of samples serves to limit the region over which samples are obtained, the RCE classifier might provide a similar restriction with its hypersphere boundaries. The generalization component of RCE-2 was modified to test this approach. The resulting algorithm, named RCE-3 (described in Appendix B), is compared with its predecessor RCE-2 in Table 5.

The performance of the nearest-neighbor classifier indicates that these databases are not suited for improved generalization by increasing the number of nearest neighbors from 1 to 3. Both the Iris and Sonar databases register performance deficits; the Voting database alone shows a modest improvement. The relationship between the performance of RCE-2 and RCE-3 mirrors that of its nearest-neighbor counterparts, suggesting that hyperspheres managed by RCE-3's training algorithm are insufficient to limit nonlocal influence at generalization.

Rejection of Patterns from an Unknown Class

One proposed advantage of hyperspherical classifiers over nearest-neighbor classifiers is the ability to recognize patterns from an unknown class as not belonging to any class known to the classifier, a point made by Li (1988: 21) but not tested. Therefore, experiments were conducted with the Iris database to test whether these RCE classifiers can reject patterns from an unknown class. All trials were performed with the training instances used in previous experiments, but with all instances from one of the three classes removed in turn. Trials were performed with DIST_PCT set at 55, 75, and 95% and for algorithm RCE-2 also at 35%. Unlike previous experiments, values obtained are the result of 50 trials. Results from RCE-1 and RCE-2 are shown in Table 6.

The values for the classification of unknown instances shown in Table 6

TABLE 5. Correct Classification of New Instances

	Correct classification (%)			
Classifier	Iris database	Sonar database	Voting database	Weighted average
Nearest-neighbor (k_g = 1)	96.0	95.2	91.2	93.0
Nearest-neighbor (k_g = 3)	94.0	85.6	94.0	91.6
RCE-2	95.0 ± 2.4	91.8 ± 2.2	89.2 ± 2.0	90.7
RCE-3	94.2 ± 2.4	82.0 ± 3.7	90.0 ± 2.5	88.3

Plurality voting at generalization (RCE-3) compared with class assignment of nearest enclosing hypersphere (RCE-2). Nearest-neighbor classifiers are provided for comparison.

TABLE 6. Detection of Instances from an Unknown Class in the Test Set for Each of the Three Classes in the Iris Database

	Instances (%)					
Class removed:	Iris setosa		Iris versicolour		Iris virginica	
Classifier tested:	RCE-1	RCE-2	RCE-1	RCE-2	RCE-1	RCE-2
Correct detection of instances from unknown class	100 ± 0.0	70.5 ± 43	8.0 ± 8.3	6.4 ± 17	18.9 ± 18	16.7 ± 24
Correct classification of instances from known classes	85.5 ± 7.5	86.3 ± 10	100 ± 0.0	99.4 ± 2.3	99.4 ± 2.0	95.4 ± 2.0

Also shown is the classifier's ability to classify patterns from the two known classes in the test set simultaneously and correctly.

were chosen as the best over the range of DIST_PCT values tested. For different classes removed from the training set, different values of DIST_PCT provided the best rejection of instances from the unknown class in the test set.

Table 6 demonstrates that an RCE classifier's labeling a test instance as not belonging to a known class is unpredictable. Only the performance of RCE-1 with the removal of instances from the class Iris setosa provides acceptable detection of the unknown class, but it then correctly classifies only 85% of the test instances from the remaining known classes. Removal of each class in turn does not compromise recognition of patterns from the known classes, but detection of instances from the unknown class never exceeds 20% of the test instances from the unknown class.

This behavior seems an inescapable consequence of the fundamental characteristics of the RCE classifier. Because the RCE classifier's training algorithm initializes hyperspheres with large radii and shrinks them only when necessary, the RCE classifier tends to overgeneralize. Hyperspheres will never be smaller than is necessary to classify all the training instances correctly. But the same hyperspheres may be so large that they cover regions of the instance space occupied by patterns from an unknown class. Hence, the ability to classify a new pattern as an instance of an unknown class depends critically on the orientation of the unknown class to the known ones. If, for example, the unknown class lies between two known classes in the instance space, it is impossible that instances of the unknown class will be classified as "unknown." Hyperspheres of instances from the known classes on either side will overlap the region of unknown class and be bounded only by instances from the other known class. Within the hypersphere paradigm, hypersphere modification by positive reinforcement, as done by Batchelor (1974), seems the only solution. This, however, confronts us with a training procedure not guaranteed to converge to its error criterion.

SUMMARY

The significant aspects of this paper are now summarized.

1. Restricted Coulomb energy classifiers (RCE), as typically implemented (Lee, 1989; Li, 1988; Scofield et al., 1988), are related to hyperspherical classifiers as defined by Batchelor (1974).
2. Hyperspherical classifiers (Batchelor, 1974) are similar to "prototype models" of concept formation from cognitive psychology.
3. RCE classifiers (Scofield et al., 1988) are similar to "exemplar models" of concept formation from cognitive psychology.
4. Simple RCE classifiers [e.g., DSND (Li, 1988)] are competitive with well-known classifiers in their ability to generalize, but their performance is at the lower end of the range.
5. Modifications to the simple RCE classifier DSND (Li, 1988) that produce RCE-2 increase average correct classification of test instances from 83.2 to 90.7%, an absolute increase of 7.5 percentage points and a relative improvement in performance of 9.0% over that of DSND. The modifications to DSND do not entail significant differences in computational costs.
6. RCE-2, the best-performing RCE classifier (in terms of accurate classification of test instances), provides no advantage over its underlying incremental neatest-neighbor component INN in terms of average generalization accuracy, variation in generalization accuracy, average storage requirements, or average number of training epochs.
7. The nearest-neighbor classifier ($k_g = 1$) correctly classified test instances better than any other classifier examined.
8. The incremental nearest-neighbor classifier INN performed second to the nearest-neighbor classifier in correct classification of test instances and at a substantial reduction in storage. INN is identical to IB2 (Aha et al., 1991), except for the choice of distance metric. The interested reader is referred to this work for a thorough examination of the classifier.

9. Generalizing RCE-2 in the manner of the nearest-neighbor classifier (i.e., increasing the number of nearest neighbors from 1 to 3) mirrors the performance trend of the nearest-neighbor classifier.
10. Detection of instances from an unknown class is possible with an RCE classifier, but performance is unpredictable, typically unsatisfactory, and subject to the characteristics of the instance database.

DISCUSSION

The experiments reveal no advantage to using a simple RCE classifier such as DSND or RCE-2 in lieu of the underlying incremental nearest-neighbor component INN. Nevertheless, the possibility still exists that an RCE classifier can be built that, on average, performs better than its incremental nearest-neighbor component. One approach was presented in which RCE-2, the RCE classifier whose hyperspheres are least effective among the RCE classifiers tested, was modified to perform generalization with $k_g = 3$ instead of $k_g = 1$. The results were equivocal but consistent with the performance of the nearest-neighbor classifier ($k_g = 3$). These results indicate that hyperspheres in RCE-3 are incapable of preventing nonlocal influence at generalization in a small database. An interesting experiment would compare the modification of DSND for $k_g = 3$ with the original DSND, RCE-2, RCE-3, INN ($k_g = 1$ and 3) and nearest-neighbor ($k_g = 1$ and 3) classifier. Hyperspheres in DSND exercise more influence at generalization than hyperspheres in any other RCE classifier tested and thus have the best chance of preventing nonlocal interference at generalization.

Another approach to improved generalization, suggested by Scofield et al. (1988), associates frequency counts with stored points in an attempt to approximate the probability densities of the distributions. A similar approach for instance-based classifiers has been taken by Aha et al. (1991) in their IB3 algorithm. Comparison of IB3 with its RCE counterpart would make an interesting study, especially in the application of these classifiers to noisy data.

Although a hyperspherical classifier can, in theory, detect when an instance does not belong to a known category, the experiments conducted here indicate that simple RCE classifiers cannot perform this function with a high degree of confidence. Perhaps the more general hyperspherical classifier (Batchelor, 1974) using positive feedback and relocation of stored patterns can more effectively recognize instances from an unknown class. But this brings forth the possibility of computationally intensive training procedures—even ones no longer guaranteed to converge to their error criterion.

Hyperspherical classifiers have been investigated since the 1960s. RCE classifiers have been discussed in the literature since Scofield et al. (1988) and previously existed within the conceptual framework described by Reilly et al. (1982). Until the present study the role of hyperspheres in the classification process had not been examined. As hyperspheres in several RCE classifiers have now been shown to provide no value, I think it incumbent upon future developers of hyperspherical classifiers to demonstrate that the hyperspheres in their classifiers positively contribute to their classifier's performance.

ACKNOWLEDGMENTS

I wish to thank the reviewers for many suggestions that contributed to the clarity and accuracy of this article. Thanks also to Kenneth R. Anderson, George M. Chaikin, Martin S. Glassman, Narendra K. Gupta, Stephen J. Hanson, Steven W. Norton, and Raymond L. Watrous for their comments on early drafts of this work. Marianne Steenken wrote much of the FORTRAN code for the RCE and nearest-neighbor classifiers.

APPENDIX A: DATA

Experiments were performed with the three databases described below.

1. Iris database (Fisher 1936)

Availability:	University of California at Irvine
Number of classes:	3
Training (100 instances):	33% Iris virginica, 33% Iris versicolour, 34% Iris setosa
Testing (50 instances):	34% Iris virginica, 34% Iris vesicolour, 32% Iris setosa
Number of attributes:	4 numeric
Missing attribute values:	None
Comments:	One of the best-known databases in the pattern recognition literature. Iris setosa is linearly separable from the other two classes; the other two are not linearly separable from each other.

2. Sonar database (Gorman and Sejnowski, 1988)

Availability:	Carnegie Mellon University
Number of classes:	2
Training (104 instances):	47.1% mines, 52.9% rocks
Testing (104 instances):	59.6% mines, 40.4% rocks
Number of attributes:	60 numeric
Missing attribute values:	None
Comments:	The database used by Gorman and Sejnowski in their study of the classification of sonar signals with a neural network. The task is to learn to distinguish between sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock.

3. Voting database (1984 U.S. Congressional voting records)

Availability:	University of California at Irvine
Number of classes:	2
Training (218 instances):	62.8% Democrat, 37.2% Republican
Testing (217 instances):	59.9% Democrat, 40.1% Republican
Number of attributes:	16 binary (Yea, Nay)
Missing attribute values:	Several (each treated as the third value: "Unknown")
Comments:	Includes votes for each member of the U.S. House of Representatives on the 16 key votes identified by the *Congressional Quarterly Almanac*. Because the classifiers require numeric features, the following substitutions were made: -1 for Nay, 1 for Yea, 0 for Unknown. The task is to learn to predict party affiliation from voting action.

APPENDIX B: TRAINING AND GENERALIZATION ALGORITHMS

Identifiers

`CP1`: Closest pattern (city-block distance using integer features) in the classifier to `TR_PAT(I)`.

`CP2`: `CP3(1)`: closest pattern (city-block distance using integer features) in the classifier to `TR_PAT(I)` whose hypersphere encloses `TR_PAT(I)`.

`CP3( )`: All patterns in the classifier whose hyperspheres enclose `TR_PAT(I)` ordered by increasing city-block distance from `TR_PAT(I)`.

`DIST_PCT`: User-selected parameter (.01–.99) for setting a stored pattern's radius as the percentage of the distance to the nearest stored pattern of a different class.

k_g: Number of nearest neighbors returned per test instance at generalization.

k_t: Number of nearest neighbors returned per training instance during training.

`TR_PAT(I)`: *I*th pattern (instance) in the training set.

Functions

`class( )`: Returns the class of its argument.
`class_of_majority( )`: Returns class of majority of its arguments.
`dist(x,y)`: Returns to city-block distance between its arguments.
`min(x,y)`: Returns lesser of its arguments.
`radius( )`: Addresses the radius of its argument.

Notes

1. Italicized remarks are high-level descriptions of complex operations.
2. Test enclosed in braces { } is a comment.
3. The value 256 used throughout the algorithms represents an implementation restriction on each pattern feature's maximum number of distinct values.

DSND {Disjoint Spheres/No Drift, from Li (1988: 62)}

Training algorithm: {Summary: Modify all hyperspheres of patterns from other classes whose hyperspheres enclose the new training instance.}

```
do until ( All training instances are correctly classified and no changes
    are made to the classifier during a training epoch )
  for i := 1 step 1 until Number of training patterns do
    Return N patterns (CP3(k), k ≤ N) ordered by increasing distance from
    TR_PAT(i) whose hyperspheres enclose TR_PAT(i).
    if ( N > 0 ) then
      if ( All patterns returned are from the same class ) then
        if ( class(TR_PAT(i)) ≠ class(CP2) ) then
          Store TR_PAT(i) into classifier.
          radius(TR_PAT(i)) := DIST_PCT × dist( TR_PAT(i), CP2 )
          for j := 1 step 1 until N do
            radius(CP3(j)) := DIST_PCT × dist( TR_PAT(i), CP3(j) )
          endfor
        endif
      else {All patterns returned are not from the same class}
        for j := 1 step 1 until N do
          if ( class(TR_PAT(i)) ≠ class(CP3(j)) ) then
            radius(CP3(j)) := DIST_PCT × dist( TR_PAT(i), CP3(j) )
          endif
        endfor
        for j := 1 step 1 until N do
          if ( class(TR_PAT(i)) ≠ CP3(j) ) then
            Store TR_PAT(i) into classifier.
            radius(TR_PAT(i)) := DIST_PCT × dist( TR_PAT(i), CP3(j) )
            stop
          endif
        endfor
      endif
    else {N=0}
      Store TR_PAT(i) into classifier.
      radius(TR_PAT(i)) := min( DIST_PCT ×dist( TR_PAT(i), Nearest stored
      pattern whose class is not equal to class(TR_PAT(i)) ),
      ( Number of features in each pattern ×256 ) )
    endif
  endfor
enddo
```

Generalization algorithm:

```
Return class( Closest stored pattern of an enclosing hypersphere )
```

RCE-1 {Modified from DSND}

Training algorithm: {Summary: If all hyperspheres enclosing the training instance are from the same incorrect class, then decrease all of these hyperspheres. If not all hyperspheres enclosing the training instance are from the same incorrect class, but the closest stored pattern whose hypersphere encloses the training instance is from the wrong class, then reduce this hypersphere.}

```
do until ( All training instances are correctly classified and no changes
    are made to the classifier during a training epoch )
  for i := 1 step 1 until Number of training patterns do
    Return N patterns (CP3(k), k ≤ N) ordered by increasing distance from
      TR_PAT(i) whose hyperspheres enclose TR_PAT(i).
    if ( N > 0 ) then
      if ( All patterns returned are from one class ) then
        if ( class(TR_PAT(i)) ≠ class(CP2) ) then
          Store TR_PAT(i) into classifier.
          radius(TR_PAT(i)) := DIST_PCT × dist( TR_PAT(i), CP2 )
          for j := 1 step 1 until N do
            radius(CP3(j)) := DIST_PCT × dist( TR_PAT(i), CP3(j) )
      endfor
    endif
  else { All patterns not from one class }
    if ( class(TR_PAT(i)) ≠ class(CP2) ) then
      Store TR_PAT(i) into classifier.
      radius(CP2) := DIST_PCT × dist( TR_PAT(i), CP2 )
      radius(TR_PAT(i)) := DIST_PCT × dist( TR_PAT(i), CP2 )
    endif
  endif

    else {N=0}
      Store TR_PAT(i) into classifier.
      radius(TR_PAT(i)) := min( DIST_PCT ×dist(TR_PAT(i), Nearest stored
        pattern whose class is not equal to class(TR_PAT(i)),
      ( Number of features in each pattern ×256 ) )
    endif
  endfor
enddo
```

Generalization algorithm: Generalization algorithm for DSND.

RCE-2 {Modified from RCE-1}

Training algorithm. {Summary: Modify only the hypersphere of the closest stored pattern from a different class.}

```
do until ( All training instances are correctly classified and no changes
    are made to the classifier during a training epoch )
  for i := 1 step 1 until Number of training patterns do
    if ( CP2 exists for TR_PAT(i) ) then
      if ( class(TR_PAT(i)) ≠ class(CP2) ) then
        Store TR_PAT(i) into classifier.
      radius(TR_PAT(i)) := DIST_PCT × dist( TR_PAT(i), CP2 )
      radius(CP2) := DIST_PCT × dist( TR_PAT(i), CP2 )
    endif
  else
      Store TR_PAT(i) into classifier.
      radius(TR_PAT(i)) :=  min( DIST_PCT ×dist( TR_PAT(i), Nearest stored
        pattern whose class is not equal to class(TR_PAT(i)) ),
        ( Number of features in each pattern ×256 ) )
 endfor
enddo
```

Generalization algorithm: Generalization algorithm for DSND.

RCE-3

Training algorithm: Training algorithm for RCE-2.

Generalization algorithm:

```
if ( Number of classes of training instances < 3 ) then
  Recall maximum of 3 closest patterns whose hyperspheres enclose the test
    pattern.
else
  Recall all patterns whose hyperspheres enclose the test pattern.
endif
if ( Plurality of patterns returned belong to one class C ) then
  Return C.
else
  Return class( Closest pattern of an enclosing hypersphere )
endif
```

INN {Incremental Nearest-Neighbor Classifier}

Training algorithm:

```
do until ( All training instances are correctly classified and no changes
   are made to the classifier during a training epoch )
  for i := 1 step 1 until Number of training patterns do
   if( class(TR_PAT(i)) ≠ class(CP1) or classifier contains no patterns )
    then
      Store TR_PAT(i) into classifier.
   endif
  endfor
enddo
```

Generalization algorithm:

```
Return class( Nearest stored pattern to test pattern )
```

REFERENCES

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

Alder, H. L., and E. B. Roessler. 1972. *Introduction to Probability and Statistics.* San Francisco: W. H. Freeman.

Bachmann, C. M., L. N. Cooper, A. Dembo, and O. Zeitouni. 1987. A Relaxation Model for Memory with High Storage Density. *Proc. Natl. Acad. Sci. USA* 84:7529–7531.

Batchelor, B. G., and B. R. Wilkins. 1968. Adaptive Discriminant Functions. *Pattern Recogn. I.E.E.E. Conf. Publ.* 42:168–178.

Batchelor, B. G. 1969. Learning Machines for Pattern Recognition. Ph.D. thesis, University of Southampton, Southampton, England.

Batchelor, B. G. 1974. *Practical Approach to Pattern Classification.* New York: Plenum.

Batchelor, B. G. 1978. Classification and Data Analysis in Vector Space. In *Pattern Recognition,* ed. B. G. Batchelor, pp. 67–116. London: Plenum.

Cooper, P. W. 1962. The Hypersphere in Pattern Recognition. *Inform. Control* 5:324–346.

Cooper, P. W. 1966. A Note on an Adaptive Hypersphere Decision Boundary. *IEEE Trans. Electron. Comput.* 948–949.

Dembo, A., and O. Zeitouni. 1988. General Potential Surfaces and Neural Networks. *Phys. Rev. A* 37(6):2134–2143.

Dowdy, S., and S. Wearden. 1991. *Statistics for Research,* 2nd ed. New York: Wiley.

Duda, R. O., and P. E. Hart. 1973. *Pattern Classification and Scene Analysis.* New York: Wiley.

Fahlman, S. E. 1989. Faster-Learning Variations on Back-Propagation: An Empirical Study. In *Proceedings of the 1988 Connectionist Models Summer School,* eds. D. S. Touretzky, G. Hinton, and T. Sejnowski, pp. 38–51. Palo Alto, CA: Morgan Kaufmann.

Fahlman, S. E., and C. Lebiere. 1990. The Cascade-Correlation Learning Architecture. Technical Report: CMU-CS-90-100, School of Computer Science, Carnegie Mellon University, Pittsburgh.

Fisher, R. A. 1936. The Use of Multiple Measurements in Taxonomic Problems. *Annu. Eugenics* 7(2):179–188.

Flynn, M. J., R. Zeidman, and E. Lochner. 1988. Sparse Distributed Memory Prototype: Address Module Hardware Guide. Technical Report CSL-TR88-373, Stanford University Computer Systems Laboratory, Palo Alto, CA.

Gorman, R. P., and T. J. Sejnowski. 1988. Analysis of Hidden Units in a Layered Network Trained to Classify Sonar Targets. *Neural Networks* 1:75–89.

Hopfield, J. J. 1982. Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proc. Natl. Acad. Sci. USA* 79:2554–2558.

Kanerva, P. 1988. *Sparse Distributed Memory.* Cambridge, MA: MIT Press.

Lee, Y. 1989. Classifiers: Adaptive Modules in Pattern Recognition Systems. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge.

Lee, Y., and R. P. Lippmann. 1990. Practical Characteristics of Neural Network and Conventional Pattern Classifiers on Artificial and Speech Problems. In *Advances in Neural Information Processing Systems 2,* ed. D. S. Touretzky, pp. 168–177, Palo Alto, CA: Morgan Kaufmann.

Li, R. 1988. An Experimental Approach to Module Diagnostics Using an Associative Memory. Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge.

Matheus, C. J. 1987. Conceptual Purpose: Implications for Representation and Learning in Machines and Humans. Report No. UIUCDCS-R-87-1370, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana.

Medin, D. L., and M. M. Schaffer. 1978. Context Theory of Classification Learning. *Psychol. Rev.* 85(3):207–238.

Medin, D. L., and E. E. Smith. 1984. Concepts and Concept Formation. *Ann. Rev. Psychol.* 35:113–138.

Moody, J., and C. Darken. 1989. Learning with Localized Receptive Fields. In

Proceedings of the 1988 Connectionist Models Summer School, eds. D. S. Touretzky, G. Hinton, and T. Sejnowski, pp. 133–143. Palo Alto, CA: Morgan Kaufmann.

Nowlan, S. J. 1990. Maximum Likelihood Competitive Learning. In *Advances in Neural Information Processing Systems 2,* ed. D. S. Touretzky, pp. 574–582. Palo Alto, CA: Morgan Kaufmann.

Posner, M. I., and S. W. Keele. 1970. Retention of Abstract Ideas. *J. Exp. Psychol.* 83:304–308.

Potter, T. W. 1987. Storing and retrieving data in a parallel distributed memory system. Ph.D. dissertation, Department of Systems Science, T. J. Watson School of Engineering, Applied Science and Technology, State University of New York, Binghamton.

Quinlan, J. R. 1986. Induction on Decision Trees. *Machine Learning* 1:81–106.

Reed, S. K. 1972. Pattern Recognition and Categorization. *Cogn. Psychol.* 3:382–407.

Reilly, D. L., L. N. Cooper, and C. Elbaum. 1982. A Neural Model for Category Learning. *Biol. Cybernet.* 45:35–41.

Reilly, D. L., C. Scofield, C. Elbaum, and L. N. Cooper. 1987. Learning System Architectures Composed of Multiple Learning Modules. *Proc. IEEE First International Conference on Neural Networks,* San Diego, June 21–24.

Rimey, R., P. Gouin, C. Scofield, and D. L. Reilly. 1986. Real-Time 3-D Object Classification Using a Learning System. *Intelligent Robots and Computer Vision Proc. SPIE* 726:552–557.

Sachs, L. 1982. *Applied Statistics,* English translation by Z. Reynarowych. New York: Springer-Verlag.

Scofield, C. L., D. L. Reilly, C. Elbaum, and L. N. Cooper. 1988. Pattern Class Degeneracy in an Unrestricted Storage Density Memory. In *Neural Information Processing Systems,* ed. D. Z. Anderson, pp. 674–682. New York: American Institute of Physics.

Requests for reprints should be sent to Michael J. Hudak.