A RECURSIVE HYPERSPHERIC CLASSIFICATION ALGORITHM

Salyer B. Reed
Carl G. Looney
Sergiu M. Dascalu
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada 89557, USA
{sreed, looney, dascalus}@cse.unr.edu

Abstract

This paper presents a novel method for learning from a labeled dataset to accurately classify unknown data. The recursive algorithm, termed Recursive Hyperspheric Classification, or RHC, can accurately learn the classes of a labeled, n-dimensional dataset via a training method that recursively spawns a set of hyperspheres, endeavoring to separate and divide the feature space into partitions. This produces a comprehensive mapping of the space. These hyperspheres provide guidance for the search because they are recursively traversed. Some benchmarking has been performed on various data sets and has shown to yield superior results to more traditional artificial methods.

Keywords: Classification, recursive hyperspheres, center of gravity, RHC

1 INTRODUCTION

Classification, a subset of machine learning, is the systematic process of partitioning a set of feature vectors into classes. Clustering [8, 10] is the most common way of doing this. However, once a set of feature vectors has been classified, it can be used to recognize unknown feature vectors, that is, to assign them to their correct classes using information from the known classes. This recognition process is sometimes also called classification of the new unknown feature vectors by inferring their classes based on rules or associations from the original set of known (labeled) feature vectors.

The *Recursive Hyperspheric Classification* (RHC) method proposed here is an abstract and recursive classification method used to obtain domain knowledge of a labeled set of feature vectors and then provide accurate inferences as to the class of each new unknown feature vector.

A plethora of classification techniques and algorithms currently exist, including many regressive methods such as artificial neural networks, fuzzy logic, and support vector machines. Artificial neural networks are ubiquitous in the machine learning community as they are used extensively [9]. Since their conception, neural networks have been useful in recognition processes and compare favorably to all methods, including statistical models [1]. Many popular neural networks (NNs), including backpropagation and radial basis function neural networks, are systems that embed knowledge as weights gleaned from training on a known (labeled) set of feature vectors. They then map unknown feature vectors into an output label.

Fuzzy neural networks (FNNs) use fuzzy rules that are learned during training [2], but another type is similar to probabilistic neural networks (PNNs) [6, 7] in that they use fuzzy set theory in place of probability theory. Genetic algorithms, too, have been used in conjunction with neural networks to try to increase reliability and accuracy [3], but they are relatively slow to learn and rather non-optimal when compared with radial basis function NNs that optimize weight learning.

Support vector machines (SVMs) are very popular today [11] although the learning process optimizes a Lagrangian function and is very complicated. Also, SVMs are hard to outperform when there are two or three classes. Fuzzy support vector machines were defined in [5] in 2002. Support vector based fuzzy neural networks are slightly more accurate than SVMs, at least on certain data sets [4].

This paper examines a novel, fast method for learning from a labeled dataset to classify (recognize) unknown, i.e., unlabeled data. The RHC algorithm ensures fast classification as well accurate results.

In short, the RHC is a set of hyperspheres strategically spawned in a dimensional space for the purpose of classification. The system of hyperspheres, once created, makes inferences from the known classes to the unknown inputs by traversing the hierarchal structure of those hyperspheres.

The remaining sections describe the RHC algorithm and its applications. In Section 2, the RHC algorithm is discussed in detail, including the characteristics and production of hyperspheres. Section 3 describes the results, where benchmarking and validation of the algorithm are performed using two popular datasets. Finally, a brief discussion about the RHC algorithm is included, followed by the conclusions and future work.

2 THE ALGORITHM

It is the endeavor of the RHC algorithm to accurately and precisely perform classification on various, complex multivariate datasets. Once the classes are separated and identified, validation and verification may be performed.

2.1 Algorithm Terminology

In RHC, classification is achieved by first constructing a collection of hyperspheres. Here, each hypersphere has a designated focal point, i.e., center of gravity, and a radius [Figure 1].

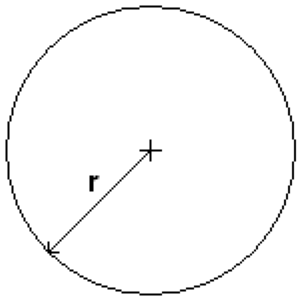

Figure 1. A hypersphere in two dimensions.

Furthermore, every such hypersphere possesses two additional properties: 1) a class designation; and 2) a set of spawned hyperspheres. Essentially, every hypersphere in the search space is a descriptive entity, for the hypersphere provides direction in search, to be discussed in later sections.

2.2 Initialization

Prior to training, it is imperative that the classification for each training vector be known. It is also beneficial to know the dimensional boundaries of the space being surveyed and probed by obtaining a diverse and broad set of input vectors for training.

2.3 Creating the First Hypersphere

After the initial acquisition of a multitude of input vectors, the derivation of the first hypersphere commences. It is imperative the first hypersphere encapsulate all input vectors of the training set. As this hypersphere encompasses all vectors in the dimensional space, each vector resides within the bounding radius of this hypersphere. The location of the center of gravity ($\overline{COG}$) is defined as follows:

$$\overline{COG} = \frac{\sum \overline{v_i}}{n},$$

where $\overline{v_i}$ is vector i, and n is the number of vectors in the dataset.

Following the determination of the center of gravity, the radius of the hypersphere must be defined. As mentioned, all input vectors must reside within the bounds of the first hypersphere. Therefore, the radius is defined as the minimum radius that encompasses all vectors in the sample set, which is also the Euclidean Distance of the farthest positioned vector from the COG [Figure 2].

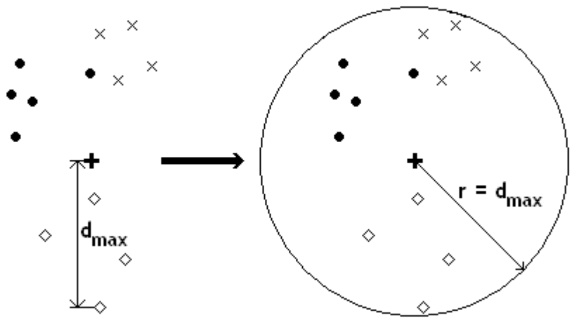

Figure 2. Creating a radius about the first COG.

Finally, the hypersphere must be assigned a class. Defining the hypersphere's class designation is rather mundane and simplistic. The heuristic exploited in RHC assigns the hypersphere's class to be that of the outermost vector, which is the vector previously used in determining the radius of the hypersphere.

2.4 Creating Hyperspheres

As indicated, the first hypersphere envelops the entire dataset of defined space. Therefore, every sample vector is enclosed within this radius. However, having a solitary hypersphere provides little insight into the taxonomy of the data. It is the objective of RHC to partition the sample space, and to accomplish this task RHC spawns additional hyperspheres.

During each iteration, every hypersphere in the space will potentially spawn additional hyperspheres. When a hypersphere spawns, or produces, an additional hypersphere, the offspring is said to be a *child* of the *parent* hypersphere. This is representative of a tree data structure, so similar terminology is employed.

The process of spawning begins here: for each iteration, for all the vectors residing within the bounds of the hypersphere but not within the radii of the hypersphere's spawn, separate them into sets according to their classification. Next, for each set compare the set's class to that of the hypersphere. If the two classes are dissimilar, proceed to spawn a new hypersphere from that group of vectors. The algorithm, in pseudocode form, for spawning new children, is described in Figure 3.

```
Spawn(Sphere)
{
  FOR EACH Child IN Sphere.Children
    Spawn(Child)
  ENDFOR

  FOR EACH Vector IN Sphere.Radius
    IF Vector.Class DOES NOT EQUAL
Sphere.Class
      IF Vector NOT IN ANY Sphere.Children
        MultiMap.Add(Vector.Class, Vector)
      ENDIF
    ENDIF
  ENDFOR

  FOR EACH Key IN MultiMap
    Child = CreateCentroid(MultiMap.Key.Values)
    Sphere.Children.Add(Child)
  ENDFOR
}
```

Figure 3. Spawning algorithm.

When spawning a new child, the center of gravity for the newly spawned child should be calculated using the previous COG equation. On the other hand, the radius of the spawn is given to be:

$$r_{spawn} = r_{parent} - \left\| c_{spawn} - c_{parent} \right\|,$$

where r_{spawn} is the radius of the spawn, r_{parent} is the radius of the parent, and $\|c_{spawn} - c_{parent}\|$ is the Euclidean Distance between the COG of the child and the COG of the parent. It is noted that the redundant process of spawning hyperspheres generates an assortment of hyperspheres with diminishing radii [Figure 4].

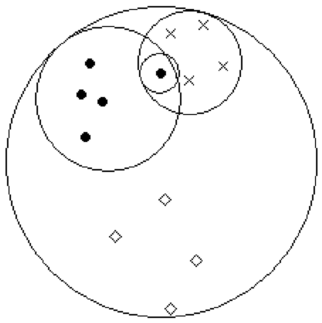

Figure 4. Spawned hyperspheres.

2.5 The Stopping Condition

The spawning of hyperspheres is a recursive process. At each new epoch a child is spawned from a set of vectors with similar classification, not already encompassed by a child. Training is complete when every hypersphere in the dimensional space does not spawn an additional hypersphere. Once training is concluded, the search space is completely described by the hyperspheres.

3 CATEGORIZATION

Following a complete RHC description of the dimensional space, one may use the collective set of hyperspheres for categorization.

3.1 Categorical Terminology

Two or more hyperspheres are regarded as being *independent* if they are not direct descendants of one another; hyperspheres are *dependent* if the converse is true.

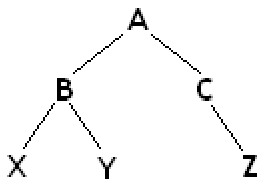

Figure 5. The hierarchy of hyperspheres.

To illustrate this principle, in Figure 5, all hyperspheres are dependent upon Hypersphere A. Hyperspheres X and Y are also dependent upon B but are independent of each other. Hyperspheres B and Z are independent as well as hyperspheres B and C.

3.2 Mapping

Classifying new, unknown vectors is also a recursive process. The system created by RHC assigns a class label for each input vector. To produce this label the vector is presented to the system. Starting with the first hypersphere, the vector is checked for proximity. If the vector resides within the hypersphere, the vector is checked against all the hypersphere’s spawn. For each spawn that encapsulates the vector, the vector is compared to that spawn’s spawn. Eventually, if no other spawn can describe, or enclose, the vector, the vector assumes the class of the last hypersphere that encloses it.

In the case a vector is described, or enclosed, by two or more *independent* hyperspheres and their respective radii, a comparison is made in the list of candidates. In all, the hypersphere with the smallest radius is regarded as the hypersphere most like the vector, and so the vector is assigned to the class of this hypersphere.

This navigation over the set of hyperspheres is synonymous to a tree traversal. Starting from the root, or first hypersphere, a search is performed on the children. If the unlabeled vector resides within the boundaries of a child, the child, too, is parsed. For all children not encompassing the unlabeled vector, they are trimmed, or excluded.

It is noted that each child has a smaller radius than its parent. This contraction facilitates defining and constraining the search space. Ultimately, it is the hyperspheres' boundaries, or radii, that partition the space into classes.

4 RESULTS AND BENCHMARKING

RHC was evaluated for accuracy and performance. In a comparison with many complex, yet highly accurate, algorithms used in classification, RHC touts exceptional speed and comparable performance.

4.1 Wisconsin Breast Cancer Dataset

Training and validation for RHC was performed on the Wisconsin Breast Cancer Dataset, which is a very popular multivariate dataset widely employed by the machine learning academia. The dataset is a compilation, or repository, of tissue features, containing 699 feature vectors; however, as sixteen vectors are incomplete, or missing a feature value, the incomplete vectors are removed, leaving 683 feature vectors. The features include: clump thickness, uniformity of cell size, uniformity of cell shape, marginal adhesion, single epithelial cell size, bare nuclei, bland chromatin, normal nucleoli, and mitoses. The first feature, the ID, is discarded, for the feature provides no statistical significance. Therefore, only nine of the features were used in training and validation. Finally, associated with each vector is a single classification: malignant or benign.

When testing the dataset, it is partitioned into two subsets: a training set and a validation set. The training set is first presented to the RHC algorithm. As mentioned, the RHC spawns a multitude of hyperspheres, which describe the space, from the training set. Once all hyperspheres are spawned, validation is performed using the validation set. It is here that the remaining vectors from the set are introduced to the system for classification.

EHC, when using the Wisconsin Breast Cancer Dataset, was benchmarked using GANN, or Genetic Algorithm and Neural Network [3]; OSRE, or Orthogonal Search-Based Rule Extraction [12]; a SVM, or support vector machine [13]; and KMP-mse, or Kernel Matching Pursuit Mean-Square Error [13]. The results of the evaluation are shown in Table I.

Table I. Wisconsin Breast Cancer Comparison.

Algorithm	Error Rates	Correct
OSRE	6.35%	93.65%
GANN	5.28%	94.72%
SVM	3.41%	96.59%
KMP-mse	3.40%	96.60%
RHC	4.00%	96.00%

4.2 Iris Dataset

RHC was also benchmarked using the famous Anderson's Iris Dataset. This dataset consists of 150 vectors. Each vector has four features: sepal length, sepal width, petal length, and petal width. Finally, each vector is assigned one of the three classifications: iris setosa, iris virginica, and iris versicolor.

Like the previous dataset, the iris dataset is divided into a training set and a testing set. In this benchmark, fivefold cross-validation was performed and the average error was recorded.

EHC, when using the iris dataset, was benchmarked using GFHSNN, or General Fuzzy Hyperspheroidal Neural Network [14]; GFMMNN, or General Fuzzy Min-Max Neural Network [14]; and GFNN, or General Fuzzy Neural Network [14]. The results of our comparison are shown in Table II.

Table II. Iris Algorithm Comparison.

Algorithm	Average Number of Errors	Number of Runs
GFHSNN	4.0	10
GFMMNN	3.1	8
GFNN	2.6	8
RHC	1.2	1[*]

5 DISCUSSION

It is noted that, on occasion, a vector that is presented to the system for categorization may reside outside the radii of all hyperspheres. In this instance it may hold true that the unknown vector cannot be accurately classified by the defined space and sometimes should be labeled as such. However, if the sample space is an accurate depiction of the runtime space, then the vector may reside slightly outside the bounds [Figure 6].

[*] It is noted that no matter how many runs are made, RHC will always produce the same mapping; hence, one run is sufficient.

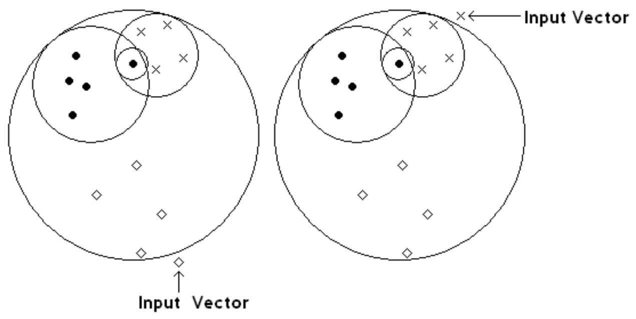

Figure 6. Vectors falling outside all radii.

In this case, considering the large distance from the vector to other hyperspheres, the vector should be considered the class of the first hypersphere. On the other hand, for a vector, which resides near the COGs of other hyperspheres, it is probable that the vector should reside within the hypersphere to the nearest COG. Later versions of RHC may take into consideration the proximity of other hyperspheres.

Also, in the improbable event that a parent spawns a child with an identical COG, resulting, too, in equal radii, the COG of the child must be nudged, or shifted, resulting in its own unique COG [Figure 7]. The consequence for not shifting includes spawning a child at each epoch so it never satisfies the stopping condition.

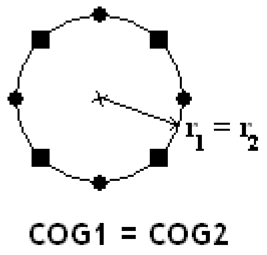

Figure 7. Two hyperspheres with the same COG.

Also, an interesting feature pertaining to RHC is no matter the geometry of the space or cluster, RHC can accurately predict the space, for RHC actively partitions the space into hyperspheres.

Finally, RHC boasts of being extensible. Many artificial intelligence and statistical methods require retraining should additions be appended to datasets. RHC, like its predecessors, can be retrained; however, it can, moreover, add additional hyperspheres of similar feature vectors to the space by determining the smallest hypersphere all vectors reside and spawn a hypersphere. To accomplish this task the vectors must reside within a small neighborhood. Once a hypersphere is created, the hypersphere must reside *fully* within another hypersphere, which is a leaf in the hierarchy [Figure 8]. Only when these conditions are satisfied may a hypersphere be appended to the hierarchical taxonomy of hyperspheres.

6 CONCLUSIONS AND FUTURE WORK

RHC possesses an uncanny ability to classify a dataset by spawning a set of hyperspheres in the dimensional space. Furthermore, classification of unlabeled data is achieved by traversing the hierarchical structure of the generated hyperspheres. As an unlabeled vector is introduced to the system, the methodical process of navigating and pruning the hierarchical taxonomy produces a list of hyperspherical candidates. Once traversal is complete and a list of candidates is available, the unlabeled vector assumes the class of the candidate with the smallest radius. During this process, it is noted that each traversal whittles away the search space, for the children's' radii are essentially partitioning the space into classes.

RHC is a simplistic, yet powerful, algorithm for classification. It can even classify rather noisy datasets. Through constant recursion and iterations, RHC spawns a set of hyperspheres that accurately map a search space. Utilizing the hyperspheres, one is able to recursively traverse them and successfully predict the classification of various inputs, or vectors, as observed from the training data.

Ultimately, continued improvements and alterations to RHC are expected, which will increase the accuracy and strength of the algorithm. For example, a special hybridized version of RHC could employ different hypergeometric shapes used in search. Another hybrid version will stretch and skew a hypersphere's radius if it resides near vectors with different classes than its own. This is synonymous to an inverse Béizer curve; vectors with classes not equal to the sphere's own class are seen as repulsive forces and distort the radius of the sphere. This, in fact, results in the hyperspheres becoming hyperblobs. Finally, further exploration of RHC in different applications is anticipated, including control systems and intelligent agents.

REFERENCES

[1] Burke, Harry B., David B. Rosen, and Phillip H. Goodman. "Comparing Artificial Neural Networks to Other Statistical Methods for Medical Outcome Prediction." *IEEE International Conference on Neural Networks*. Vol. 4, 27 June – 2 July 1994. pp. 2213 – 16.

[2] Gabrys, B., and A. Burgiela, "General Fuzzy Min-Max Neural Network for Clustering and Classification," *IEEE Transactions on Neural Networks*. Vol. 11.3, May 2000. pp. 769-83.

[3] Kermani, Bahram, Mark W. White, and H. Troy Nagle. "Feature Extraction by Genetic Algorithms for Neural Networks in Breast Cancer Classification." *IEEE 17th Annual Conference on Engineering in Medicine and Biology Society*. Vol. 1, 20 – 23 Sept. 1995. pp. 831 – 32.

[4] Lin, Chin-Teng, et al. "Support-Vector-Based Fuzzy Neural Network for Pattern Classification," *IEEE Transactions on Fuzzy Systems*. Vol. 14.1, Feb 2006. pp. 31-41.

[5] Lin, Chun-Fu, and Sheng-De Wang. "Fuzzy Support Vector Machines." *IEEE Transactions on Neural Networks*. Vol. 13.2, Mar. 2002. pp. 464-71.

[6] Looney, Carl. "A Fuzzy Neural Network with Mixed Nearest Neighbor Strategies." Technical Report, Computer Science and Engineering Department. University of Nevada, Reno. Nov. 2007.

[7] Looney, Carl, and Sergiu Dascalu. "A Simple Fuzzy Neural Network." CAINE. Nov. 2007. pp. 12-6.

[8] Looney, Carl. "Interactive Clustering and Merging with a New Fuzzy Expected Value." *Pattern Recognition*. Vol. 35, 2002. pp. 2413-23.

[9] Looney, Carl. "Pattern Recognition Using Neural Networks." New York / Oxford: Oxford University Press, 1997.

[10] Xu, Rui, and Donald Wunsch II. "Survey of Clustering Algorithms." *IEEE Transactions on Neural Networks*. Vol. 16.3, May 2005. pp. 645-78.

[11] Vapnik, Vladimir. "Statistical Learning Theory." New York: Wiley-Interscience, 1998.

[12] Etchells, T. A., and P. J. G. Lisboa. "Orthogonal Search-Based Rule Extraction (OSRE) for Trained Neural Networks: A Practical and Efficient Approach." *IEEE Transactions on Neural Networks*. Vol. 17.2, Mar. 2006. pp. 374-84.

[13] Vincent, P., and Y. Bengio. "Kernel Matching Pursuit." *Machine Learning*. 2002. pp. 169-91.

[14] Patil, P. M. and T. R. Sontakke. "A Novel Threshold Optimization of ML-CFAR Detector in Weibull Cluster Using Fuzzy-Neural Networks." *Signal Processing*. Vol. 87.9, 2007. pp. 2100-10.