HIGH DIMENSIONAL PATTERN RECOGNITION USING THE RECURSIVE HYPERSPHERIC CLASSIFICATION ALGORITHM

SALYER B. REED, TYSON R. C. REED, SERGIU M. DASCALU
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, Nevada 89557, USA
http://www.cse.unr.edu

ABSTRACT—The Recursive Hyperspheric Classification (RHC) algorithm is a novel technique that excels in classifying multivariate, labeled datasets, which may be used for identification of unknown feature vectors. When training the classifier system, RHC meticulously dissects an n-dimensional space into a taxonomic structure of classifiers, or hyperspheres. This algorithm methodically partitions the space into labeled classes. Structure and order materialize from this constant, recursive process of spawning hyperspheres; this constructs an organized hierarchical tree that, when traversed, allows labels, or classes, to be inferred from the current knowledgebase. In benchmarking, RHC boasts superior results compared to modern classification techniques. This paper offers a comprehensive examination of the RHC algorithm, including various improvements to the original version of the algorithm as well as new results of its application.

Key Word: Recursive Hyperspheric Classification, RHC, Hyperspheres, Classification

1. INTRODUCTION

Classification assigns labels to groups with similar, intrinsic properties extracted from an amassed training set encoded with a priori knowledge. This ability is inherent of any supervised learning method. Once surveyed, the compiled knowledge base provides insight into the labeling of unlabeled feature vectors, which is called pattern recognition.

This paper continues the research in [9] by improving the Recursive Hyperspheric Classification (RHC) algorithm and applying it to a new, multivariate domain. RHC is an elegant algorithm that can accurately and succinctly train on a set of classified vectors and, in turn, be used to assign labels to unlabeled data.

An exuberant number of classification algorithms are currently known to the machine learning community. Seemingly, the most popular techniques are regression models, including support vector machines (SVMs) and artificial neural networks (ANNs).

SVMs are popular in that they learn the classification and regression rules from the training data. Mapping, or transforming, the feature vectors to a higher dimensionality with the aid of a kernel, the new feature space is partitioned with a hyperplane. The margin between the hyperplane and support vectors is maximized, which results in an optimal solution. SVMs are traditionally used in binary classification, partitioning the classes with the optimal separating hyperplane. However, they show exceptional performance and versatility in multiclass classification; as such, the SVM remains extremely robust [3, 6].

Another popular, statistical technique, the Naïve Bayes Classifier, is able to classify a feature vector according to its computed posterior probability, which is the product of its feature probabilities. In computing the probabilities, the Naïve Bayesian classifier system assumes all features, or attributes, are independent; as such, the computational load is reduced due to the diminution of parameters. The system calculates each posterior probability, and the vector assumes the class that yields the greatest posterior probability. Interestingly, when the feature is continuous, the likelihood – which is a measure describing how the data fits the distribution, or model – can be represented using various density functions, including Histograms [2], Gaussian, Poisson [5], Lognormal, and Gamma functions. In practice, notwithstanding its irrational independence assumption, the Naïve Bayes Classifier – coupled with proper parameter estimation – yields satisfactory, accurate classification results.

More traditional artificial methods include artificial neural networks, which, too, exhibit a strong aptitude for classifying multivariate datasets. Traditionally, the neural network, or NN, learns the correct response to various stimuli, or input; the neural network, too, learns regression rules. A popular

implementation includes the Radial basis function neural network (RBFNN). Much like their predecessors, which include backpropagation neural networks, RBFNNs exhibit the ability to map unknown feature vectors to a feature label. However, they are unique in that the network adjusts the hidden layer, regulating the amount of nodes as well as each node's hidden function, which is usually a Gaussian function, which is unimodal. This produces an optimal solution for the network as the output, or apex, is resolved for each node. Regrettably, neural networks are slow to learn [4] and are prone to overtraining [10].

In the case of the proposed RHC algorithm, it provides a generalized, fast approach to classification. This system produces a unique classifier system, which may be used to obtain accurate inferences from the spawned, taxonomic tree. By navigating, or traversing, the tree, the nodes of the tree provide guidance in search and classification. Ultimately, it is this tree that imparts general knowledge concerning the active domain.

Subsequent sections describe the RHC algorithm in depth as well as benchmark tests. In Section 2, the RHC algorithm is explained, including the properties and spawning of hyperspheres. Section 3 outlines the process for classifying unlabeled vectors; it conveys the techniques used to traverse the classification tree. Section 4 examines the results of RHC when used to classify the wine dataset; the section is a confirmation of the algorithm's ability to classify multivariate datasets. Section 5 provides a brief discussion concerning the properties pertaining to RHC. Finally, the conclusions and future work with RHC are presented in Section 6.

2. THE ALGORITHM

Ultimately, it is the objective of any classification algorithm to accurately, precisely, and algorithmically ascertain a descriptive function of the state space that assists in categorizing unlabeled data. After generating this function, unmarked data are presented to the system, and labels are inferred from the current knowledgebase; this is pattern recognition. RHC resides in the realm of classification. It rigorously endeavors to partition the space into classes, using a hierarchical system of hyperspheres. Once assembled, the taxonomy, which can then be parsed, derives a classification label from nondescript data.

In accurately detailing the RHC algorithm, small examples pepper this section, reinforcing the reader's comprehension of this robust algorithm and its applications.

2.1 RHC Elements

A hypersphere is a geometric entity residing in an n-dimensional space, and it is the hypersphere that forms the basis for RHC. Common to every hypersphere is a set of familiar geometric properties. Each hypersphere possesses a center of gravity (COG), or centroid. Also, each hypersphere has a corresponding radius. The radius is the distance from the centroid; it defines the boundary of the hypersphere. Intrinsically, a circle is a hypersphere in a two dimensional space.

Hyperspheres, when used in conjunction with RHC, possess additional properties. First, a class label is associated with each hypersphere. These class labels provide direction in search, which is detailed later. Second, each hypersphere contains a list of children, or spawn. When a hypersphere spawns a child, the offspring is added to this list. These lists, when merged and parsed, form a tree structure, which is used to describe, or classify, the dimensional space.

2.2 Initialization

Imperative to every classification algorithm, one must obtain a diverse set of data, or input vectors, prior to classification. This variety ensures a broad spectrum of the sample space, which will minimize error, reducing misclassifications.

2.3 The First Hypersphere

Instantiating the first hypersphere proves to be a trivial task. As all input vectors are defined in the space, the COG may be computed as…

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

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

Finally, all vectors in the input dataset must reside within the hypervolume of the sphere; as such, the radius is defined to be the vector with the greatest Squared Euclidean Distance to the center of gravity, or…

$$\arg\max_{v \in SET} \left\| \overline{COG} - \overline{v} \right\|^2,$$

where $\overline{v}$ is a vector, and $\overline{COG}$ is the center of gravity of the hypersphere. This is illustrated in Figure 2.

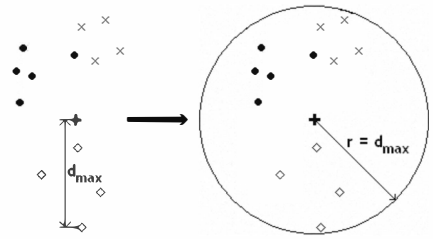

Figure 2. Defining the first hypersphere.

A full description as to why RHC uses the Squared Euclidean Distance is detailed in Section 4. Finally, as stated, each hypersphere in the space must assume a class. For simplicity and brevity, in the RHC algorithm, the first hypersphere assumes the class of the vector furthest from the center of gravity; this is the same vector used in computing the radius of the hypersphere.

2.4 Spawning Additional Hyperspheres

Concluding the creation of the first hypersphere, all vectors reside within the solitary hypersphere. However, having a single hypersphere does not provide insight into the taxonomy of the search space and its constituents. As such, it is imperative that additional hyperspheres be spawned, for these hyperspheres provide direction in search.

At the beginning of each iteration, every hypersphere in the space will potentially produce, or spawn, additional offspring. The offspring are children of the parent hypersphere. This hierarchical family structure is analogous to a tree data structure; as a result, similar vocabulary is used.

To spawn additional children, a hypersphere takes inventory of all vectors residing within its radius. For each vector in the radius of the hypersphere, separate the vectors into sets that are representative of their class; this is synonymous to creating a multimap, or hash table, where the keys are the classes and the values are sets of vectors with similar class labels. Once dissected, for each set of vectors whose key differs from the class of the hypersphere, create a new hypersphere.

To create the new hypersphere, first compute the center of gravity for all vectors in the set, or creation set. The center of gravity is guaranteed to reside in the parent; a detailed account describing this guarantee is formulated in Section 4. Once calculated, determining the radius of the child hypersphere is straightforward:

$$r_{spawn} = r_{parent} - \left\| c_{spawn} - c_{parent} \right\|^2,$$

where r_{spawn} is the radius of the spawn, r_{parent} is the radius of the parent, c_{spawn} is the COG of the spawn, and c_{parent} is the COG of the parent. This process of spawning additional hyperspheres creates a hierarchical structure of hyperspheres, where each child's radius is smaller than that of its parent; these spheres, too, are mutually tangential [Figure 3].

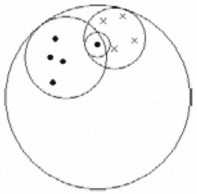

Figure 3. Spawned hyperspheres.

2.5 Stopping Condition

Training is complete when no additional hyperspheres are spawned during an iteration. Once completed, the space is considered to be partitioned and may be used in pattern recognition of unlabeled vectors.

3. PATTERN RECOGNITION

Upon completion of the dissection of the search space using hyperspheres, unlabeled vectors may be presented to the system for recognition and labeling. The hyperspheres are traversed, starting from the first hypersphere, to determine the appropriate label for the vector.

As was mentioned, the spawning of spheres creates a tree data structure. When an unlabeled feature vector is presented to the system for identification, the tree is parsed and a class is inferred.

3.1 Tree Traversal of the Hyperspheres

When an unknown feature vector is presented to the system for recognition, standard tree traversal algorithms are employed, parsing the tree at its many levels. While parsing, the algorithm will compose a list of potential candidates that can describe, or identify, the unknown vector.

Beginning with the first hypersphere, or the root of the tree, the algorithm confirms the vector is within the sphere's hypervolume. If the unlabled vector is bound by the hypersphere's radius, that hypersphere could, potentially, be a nominee for selection.

Before finalizing the hypersphere as a candidate, though, the algorithm must consider that hypersphere's children, for a child may more accurately describe the unlabeled vector; as such, the child should be considered the candidate and not the parent. Therefore, for each child of the hypersphere, the algorithm, once again, determines if the unlabeled vector can be enclosed by a child's hypervolume. If at least one child can encircle the unlabeled vector, the parent is excluded from candidacy; consequently, for each child that could, in fact, envelope the unlabeled vector, those children, too, are checked. If a child cannot enclose the unlabeled vector, it and its children are excluded from the search; this is the process of tree trimming.

On the other hand, only when – while parsing the tree – a hypersphere has no children that can encapsulate the unlabeled vector, is the hypersphere considered for candidacy; this implicitly entails the condition when a hypersphere can enclose the unlabeled vector yet contains no spawn.

From the constant recursion of tree traversal, a list of possible candidates emerges. Finally, from the list, the candidate with the smallest radius is deemed most descriptive, and the unlabeled vector assumes the class of that hypersphere.

Because each child's radius is smaller than its parent's radius, the boundaries of the search are dwindling, confining the search to smaller regions in the space. Ultimately, it is the constant recursion of spawning that partitioned the space into classes, which can actively be scrutinized to yield descriptive labels.

4. EUCLIDEAN SQUARED DISTANCE

4.1 Convexity

Earlier a statement was made guaranteeing the COG of a child hypersphere will be contained within the parent. Since the algorithm is utilizing the Squared Euclidean Distance to gauge and create hyperspheres, one must ensure the set is convex. As such, a formal proof of this fact is provided.

To show that a set S is convex, it must be shown that if $x \in S$, $y \in S$, and $a \in [0, 1]$, then $ax + (1 - a)y \in S$. The proof that sets created by the Euclidean Squared Distance are convex is as follows.

Proposition: Suppose **x** and **y** are two vectors in $\mathbb{R}^n$ and $a \in [0, 1]$, then…

$$g(a) = \sum_{i=1}^{n} \left[ax_i + (1-a)y_i\right]^2 \leq \max\left\{\sum_{i=1}^{n} x_i^{\,2}, \sum_{i=1}^{n} y_i^{\,2}\right\}$$

Proof: Note that…

$$0 \le \sum_{i=1}^{n} (x_i - y_i)^2 = \sum_{i=1}^{n} \left(x_i^{\,2} + y_i^{\,2} - 2x_i y_i \right).$$

This result follows from the fact that the sum of nonnegative numbers is nonnegative, and any number squared is nonnegative. Now examine g in further detail…

$$g(a) = \sum_{i=1}^{n} \left[ax_i + (1-a)y_i \right]^2$$

$$g(a) = \sum_{i=1}^{n} \left[a^2 x_i^{\,2} + (1-a)^2 y_i^{\,2} + 2a(1-a)x_i y_i \right]$$

$$g(a) = a^2 \sum_{i=1}^{n} \left[x_i^{\,2} + y_i^{\,2} - 2x_i y_i \right] + 2a \sum_{i=1}^{n} \left[x_i y_i - y_i^{\,2} \right] + \sum_{i=1}^{n} y_i^{\,2}$$

That is, g is a polynomial of second degree with respect to a. Since the coefficient before a^2 is positive, g is concave up. The minimum and maximum of a function occur at points where the derivative of g is zero or at the endpoints of the interval [0, 1]. If the minimum is observed to be at one endpoint, the other endpoint must be the maximum. If the minimum occurs in the interval [0, 1], then the maximum of g occurs at one of the endpoints. As such, either…

$$g(1) = \sum_{i=1}^{n} x_i^{\,2} \quad or... \quad g(0) = \sum_{i=1}^{n} y_i^{\,2}$$

is the maximum of g over [0, 1], concluding the proof.

In the context of this paper, let r be the radius determined by the process. Using the proposition, one can reason that the hypersphere, or ball…

$$B_r = \left\{ x \in R^n : \sum_{i=1}^{n} (x_i - COG_i)^2 \le r \right\}$$

is convex. That is, suppose that $x \in B_r$, $y \in B_r$, which are two points in the hypersphere, and $a \in [0, 1]$. By the proposition, the ball, B_r, is convex since…

$$\sum_{i=1}^{n} \left[ax_i + (1-a)y_i - COG_i \right]^2 = \sum_{i=1}^{n} \left[a(x_i - COG_i) + (1-a)(y_i - COG_i) \right]^2$$

$$\le \max\left\{ \sum_{i=1}^{n} (x_i - COG_i)^2, \sum_{i=1}^{n} (y_i - COG_i)^2 \right\} \le r.$$

4.2 Euclidean Distance Versus Squared Euclidean Distance

When initially conceived, RHC used Euclidean Distance when spawning and classifying; however, further refinement and iterations to the algorithm yielded improved results when using the Squared Euclidean Distance.

For example, consider two points in a two-dimensional space; the points are defined in Table I.

Table I. Comparison of Euclidean distance versus squared Euclidean distance

Vector	Dim1	Dim2
$\mathbf{v}^{(1)}$	0	0
$\mathbf{v}^{(2)}$	0	10
$\mathbf{COG}_{\mathbf{Child}}$	0	5

Method	Radius1	Radius2	Contraction Rate
Euclidean	10	5	**50%**
Sq. Euclidean	100	75	**25%**

From simple inspection, the Euclidean distance between $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$ is 10 units, and the squared Euclidean Distance is 100 units. If a child were to spawn at [0, 5], the radius of the child would be 5 units, using the Euclidean distance, for, as mentioned, the radius of the child is radius of the parent less the distance between the centroids of the two entities. Alternatively, the radius of the child would be 75 units using the squared Euclidean distance. From examination, the contraction rate is greater for the Euclidean distance. Empirical evidence shows that by slowing the contraction rate the system is better able to classify the dimensional space, for the algorithm decomposes the space at a slower rate, avoiding overshooting.

5. RESULTS

Previously, in [9] RHC has been evaluated using the Anderson's Iris Dataset and the Wisconsin Breast Cancer Dataset [1], where it has outperformed many classification algorithms and methods; however, the dimensionality of the spaces are relatively small. As such, in this work, RHC has successfully been tested and validated against the wine dataset [1], which asserts the strength and robustness of RHC on data with large dimensionality. This multivariate dataset contains 178 complete vectors. Each vector contains thirteen features: alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavanoids, nonflavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315, and praline. A single class label identifying one of three cultivars is assigned to each feature vector in the set.

When benchmarking, RHC was pitted against Fuzzy Classifier Ensemble [7], Robust Piecewise-Nonlinear M-SVM [11], and Evolutionary Computation Genetic Algorithm [8]. Prior to training and testing, the dataset is normalized, scaling all dimensional features in the range [0, 1]. While benchmarking, tenfold cross-validation was utilized. As such, the dataset was randomly partitioned into two distinct sets: one for training and one for testing and validation. Partitioning and testing for RHC was performed 1000 times, and the results of the benchmarks are presented in Table II.

Table II. Wine dataset algorithm comparison

Algorithm	Correct	Incorrect	Average Tree Height	Average Tree Nodes
RHC - Sq. Eucl. Dist.	97.90%	2.10%	6.678	20.459
RHC - Eucl. Dist.	85.60%	14.40%	3.414	55.457
EC-GA	94.80%	5.20%	- - -	- - -
Robust Piecewise-Nonlinear M-SVM	90.00%	10.00%	- - -	- - -
Fuzzy Classifier Ensemble	92.70%	7.30%	- - -	- - -

RHC's ability to actively partition the space using a common geometric shape is uncanny and results are exceptional. On average, from the results, it is apparent that RHC – when using the squared Euclidean distance – will spawn less children but have a greater tree height that than when using the Euclidean distance. This can be attributed to the contraction rate of the algorithm. Because the Euclidean distance will overshoot, it will potentially spawn additional children in the next iteration. If these new children have large enough radii, they, too, will spawn additional children at the same level. On the other hand, the squared Euclidean distance will have a greater height, for it will spawn an additional children at the next level as opposed to the current level.

6. DISCUSSION

6.1 Reduction

At times, a hypersphere may spawn a series of children that every vector in the parent can be described, or enclosed, by at least one child. In this case, the parent is considered superfluous and should be deleted. Deleting the parent includes promoting each child of the current parent to be a parent's sibling, or a child of the parent's parent. Once promoted, delete the discrepant, unnecessary hypersphere from the tree [Figure 5].

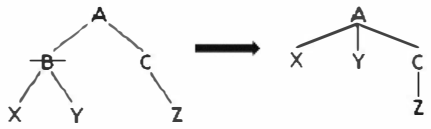

Figure 5. Deleting a parent hypersphere, promotes later hyperspheres.

6.2 Problems and Resolutions

Convergence in the RHC algorithm is guaranteed. As each child's radius is smaller than its parent's radius, the scope, or hypervolume, of each hypersphere diminishes with each new child. Again, the process of generating children is repeated until no new children are spawned. Because the volume is contracting, the number of vectors within the radius of the hypersphere, too, decreases, so given enough iterations, the algorithm will converge, and no additional children will spawn.

Unfortunately, spawning a child hypersphere, too, does not guarantee all vectors of a creation set will reside in the new child [Figure 6].

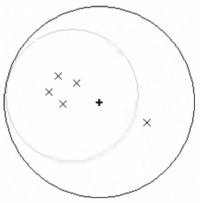

Figure 6. Of the five vectors in the hypersphere, only four are contained within the new child – the smaller circle. Next epoch, a new hypersphere will be created for the outlying vector.

If this occurs, it is common that in later epochs the excluded vectors will eventually be enclosed by an additional child hypersphere.

Also, in some scenarios, the spawned hypersphere will not contain any vector from the creation set, and it should be deleted wherefore a new set of children shall be spawned [Figure 7].

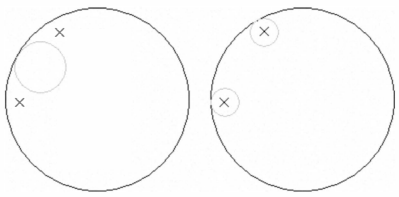

Figure 7. If a child does not contain at least one vector from the creation set, partition the creation set and create children from the new sets.

Currently, the simple heuristic employed to spawn this new set of children – called radial spawning – orders the vectors in the creation set with respect to its distance to the parent hypersphere's COG. Starting with the closest vector as the creation set, RHC spawns a hypersphere using the standard spawning method. Then, using the next closest vector in conjunction with the first vector as the creation set, RHC spawns an additional hypersphere. If this newly spawned hypersphere can enclose both vectors, RHC deletes the previously spawned hypersphere, keeping the newly created hypersphere. Thereupon, it proceeds to spawn another hypersphere using the three vectors as the creation set.

This pattern is continued until a newly spawned hypersphere cannot enclose all vectors that were used as the creation set. When this happens, instead of deleting the previously spawned child, the algorithm deletes the newly spawned child. Then, starting with the latest vector as the creation set, the radial spawning process once again invoked. Ultimately, the radial spawning method iterates over all vectors in the original creation set, creating a set of children that can, in fact, enclose all elements in the ordered list.

Finally, another improbable outcome, which nonetheless must be addressed, is the possibility that a parent spawns a child with COG equal to its own. If this condition emerges, one should translate the COG by a nominal offset, using various heuristics if needed. Another solution includes decomposing the creation set into multiple sets, where each set creates a unique hypersphere, which is synonymous to radial spawning.

7. CONCLUSION AND FUTURE WORK

RHC provides a robust classification technique that, when utilized, accurately classifies a multivariate dataset. By constantly decomposing the search space into hyperspheres, the search space is partitioned into segments, defined by the radii of the hyperspheres. Once decomposition, or training, is complete, an unlabeled feature vector may be presented to the system for classification. The methodical tactic of traversing the spawn prunes the hierarchical taxonomy of hyperspheres, endeavoring to classify the unlabeled feature vector. When traversal is complete, a winner is selected from the set of descriptive hyperspheres.

Continued improvement in RHC is expected. Of current interest is the assimilation of genetic algorithms, improving classification and reducing training overhead. RHC should harness the ability of genetic algorithms to converge on local optima when, and if, a creation set needs to be partitioned for further exploitation.

Another improvement includes a hybrid version that will stretch and skew a hypersphere's radius. If a hypersphere's radius is adjacent to vectors with a label different than its own, the radius will become transfigured. This is analogous to an inverse Béizer curve, for the radius becomes malleable and will warp because the other vectors are considered repulsive forces. This, in fact, transforms hyperspheres into hyperblobs.

Finally, other distances are to be used during the spawning process. It is known that for some datasets the Euclidean distance is not optimal; as such, other distances shall be utilized. For example, in some datasets, the Mobius distance may produce superior results.

REFERENCES

1. C. L. Blake and C. J. Merz, UCI Machine Learning Repository. University of California, Irvine. Available: http://www.ics.uci.edu/~mlearn/MLRepository.html
2. E. Elovaara and P. Myllymaki, "MDL-Based Attribute Models in Naïve Bayes Classification," *Workshop on Information Theoretic Methods in Science and Engineering*. Tampere, 2009.
3. G. M. Foody and A. Mathur, "A Relative Evaluation of Multiclass Image Classification by Support Vector Machines," *IEEE Transactions on Geoscience and Remote Sensing*. 2004, pp. 1335 – 43.
4. N. B. Karayiannis and A. N. Venetsanopolous. "Fast Learning Algorithms for Neural Networks," *IEEE Transactions on Circuits and Systems*. 1992, pp. 453 – 74.
5. S. Kim, et al, "Some Effective Techniques for Naïve Bayes Text Classification," *IEEE Transactions on Knowledge and Data Engineering*. Los Angeles, 2006, pp. 1457 – 66.
6. A. Mathur and G. M. Foody, "Multiclass and Binary SVM Classification: Implications for Training and Classification Users," *IEEE Geoscience and Remote Sensing Letters*. 2008, pp. 241 – 5.
7. T. Nakashima, G. Nakai, and Hisao Ishibuchi, "Constructing Fuzzy Ensembles for Pattern Classification Problems," *IEEE International Conference on Systems, Man, and Cybernetics*. 2003, pp. 3200 – 5.
8. M. L. Raymer, et al, "Knowledge Discovery in Medical and Biological Datasets Using a Hybrid Bayes Classifer / Evolutionary Algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*. 2003, pp. 802 – 13.
9. S. B. Reed, C. G. Looney, and S. M. Dascalu, "A Recursive Hyperspheric Classification Algorithm," *Computer and their Applications in Industry and Engineering*. 2008, pp. 156 – 60.
10. S. G. Tzafestas and Y. Anthopoulos, "A Multi-Network Architecture for High Generalization in Pattern Recognition with Back-Propagation Neural Network Modules," *IEEE International Conference on Systems, Man, and Cybernetics*. 1996, pp. 741 – 6.
11. P. Zhong and M. Fukushima, "A Regularized Nonsmooth Newton Method for Multi-class Support Vector Machines," *Optimization Methods and Software*. 2007, pp. 225 – 36.