Hypercube Neural Network Algorithm for Classification

Dominic Palmer-Brown and Chrisina Jayne

London Metropolitan University, 166-220 Holloway Road,
London N7 8DB, UK
{alfred.hofmann,ursula.barth,ingrid.haas,frank.holzwarth,d.palmer-brown@londonmet.ac.uk
anna.kramer,leonie.kunz,christine.reiss,nicole.sator,c.jayne@londonmet.ac.uk
http://www.londonmet.ac.uk/computing

Abstract. The Hypercube Neural Network Algorithm is a novel supervised method for classification. One hypercube is defined per class in the attribute space based on the training data. Each dimension of a hypercube is set to cover the full range of values in the class. The hypercube learning is therefore a rapid, one-shot form of learning. This paper presents three versions of the algorithm: hypercube without neurons; with simple neurons; and with adaptive activation function neurons. The methods are tested and evaluated on several diverse publically available data sets and compared with published results obtained on these data when using alternative methods.

Keywords: Hypercube Neural Network, Modal Learning, Adaptive Functions Neural Network, Classifictaion

1 Introduction

This paper introduces a novel supervised learning method for classification based on the idea of a hypercube and the combination of hypercubes with neural modes of supervised learning. The attraction of the hypercube is that it directly encodes and represents the range of values of across patterns in each class and on each dimension, and thereby provides immediate separation of classes wherever there is at least one non-overlapping dimensional range between the two classes. The process is extremely efficient, requiring only a single pass through the training data, very few computational steps, and only one hypercube per class. The hypercube is a radical simplification of Nested Generalised Exemplar theory [1].

The modal learning approach to neural computing [2] is combined with the hypercube mode of learning. A principle of modal learning is that each mode tackles a different aspect of the learning task, and thus the learning problem is decomposed. Rather than attempting to enhance a mode so that it becomes more complex and computationally demanding without necessarily achieving a gain in classification/learning performance sufficient to justify the increase

in complexity, the approach is to combine a selection of relatively simple and efficient modes in order to jointly solve the learning problem. The modes are combined within a single neural architecture. In this case we initially explore the hypercube mode and then its combination with the standard delta weight update learning rule for gradient descent in a single layer neural network, and then with the adaptive function neural network [3], which itself performs a modified delta rule in parallel with function adaptation.

The simplification of Nested Generalised Exemplar theory (NGE) [1] and associated hyperrectangle approaches [4] in hypercube learning avoids the NP-hard problem of determining an optimal number, location and size of the hypercubes. This simplification is achieved by stipulating one hypercube per class and allowing hypercubes to overlap. The patterns in overlapping regions are passed to the hypercube neuron to classify. The neuron is exclusively trained on patterns in the overlapping regions of the hypercube.

The nested generalised exemplar (NGE) theory is based on storing generalised examples in the form of hyperrectangles in a Euclidean n-space [1]. These hyperrectangles may overlap or nest. Each hyperrectangle has weights attached to it, which are modified during training. Once the training finishes, a test example is classified by computing the Euclidean distance between the example and each of the hyperrectangles representing the generalised exemplars. During the NGE training the number of hyperrectangles increases as training samples are presented. The existing hyperrectangle nearest to a training example expands. The performance of the NGE is compared with the k-nearest neighbour (kNN) algorithm in [4] in 11 domains and found to be significantly inferior to kNN in 9 of them. Several modifications of the NGE are suggested in [4] to improve the performance of the NGE such as avoiding overlapping hyperrectangles and batch training. Carpenter [5] et al. introduced the fuzzy ARTMAP based on fuzzy logic and adaptive resonance theory (ART). The category boxes used by the fuzzy ARTMAP with complement coding are similar to the hyperrectangles.

In this work we only require a fixed number of hypercubes which corresponds to the number of classes and a single pass through the data i.e. one shot learning; and there is no need to measure the distance from hypercube surface or hypercube volume. The problem of creating an optimal number of hyperrectangles for classification is NP-hard and several techniques have been suggested in the literature [6, 7] to reduce the number of irrelevant hyperrectangles. For example, Garcia [8] suggests the use of evolutionary algorithms to select the most influential hyperrectangles in the NGE for classification tasks. In the hypercube algorithms presented here the problem is avoided by allowing hypercubes to overlap. A neural mode of learning is used to classify the patterns in the overlapping regions of the hypercube.

Other related work includes the Fuzzy Min-Max Neural Network [9] based on a fuzzy set of hyperboxes. A fuzzy set hyperbox is an n-dimensional box defined by a min point and a max point with a corresponding membership function. The min-max points are determined using an expansion-contraction process that can learn nonlinear class boundaries in a single pass through the data and provides

the ability to incorporate new and refine existing classes without retraining. In the present work the hypercubes, like the hyperboxes, are allowed to expand and are bounded. However there is no contraction process necessary and overlapping between the hypercubes is permitted. The hypercube topology has been applied to modeling on distributed memory concurrent computers [10]. The idea of the hypercube in [10] is used to find efficient implementations of network algorithms with different connectivity patterns. It is not used for classification.

The rest of this paper is organized as follows: Section II introduces the Hypercube Neural Network (HNN) Algorithm. Section III presents the experiments, results and evaluation of the performance of the HNN. Finally, Section IV concludes the paper.

2 Hypercube Neural Network Algorithm

2.1 Hypercube without neurons

One hypercube is assigned for each class in the data set. Assume that a given training set T consists of n patterns $(x_1, \ldots, x_n)$, where x_i belongs to R^d, where d is the dimensionality of the patterns. The training patterns x_i belong m classes. Let us denote a class c made of the patterns $(x_1^c, \ldots, x_k^c)$. A hypercube h^c is defined as

$$h^c = \{y^c \in R^d, hmin_j^c <= y_j^c <= hmax_j^c, j = 1, \ldots, d\},$$

where $hmin_j^c$ and $hmax_j^c$ are the minimum and maximum values on dimension j of all patterns in class c, i.e.

$$hmin_j^c = \min(x_{1j}^c, \ldots, x_{kj}^c),$$

$$hmax_j^c = \max(x_{1j}^c, \ldots, x_{kj}^c).$$

A pattern is said to belong to a hypercube if it is inside the hypercube. If the pattern belongs to only one hypercube then it is classified according to the class label of that hypercube. If a pattern belongs to more than one hypercube the pattern is classified as belonging to the hypercube that contains the maximum number of training patterns. If the hypercube corresponds to the correct class for the pattern then the classification error is 0, otherwise 1. The performance of the classification is measured as the percentage of the correctly classified patterns. In the case of the testing set the maximum number of dimensions containing the pattern determines which hypercube the pattern belongs to.

2.2 Hypercube with simple neurons

Using the training patterns we define m hypercubes corresponding to the m classes as described in A. We associate one neuron with each hypercube h^c and denote the weight vector for that neuron with

$$w^c = (w_1^c, .., w_d^c).$$

Sample architecture of the Hypercube Neural network is shown in (Fig. 1) for a 3 class problem.

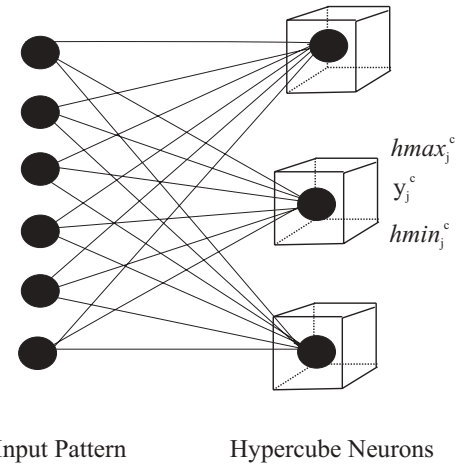

Fig. 1. Hypercube Neural Network Architecture for 3 class problem

The weight components are initialized to $\frac{1}{d}$ at the beginning of the training process. Neurons are only trained with patterns that are in overlapping regions of hypercubes. In that case, we train the hypercube neurons of all the hypercubes to which the pattern belongs. The activation for each neuron is calculated as the sum of the weighted absolute differences between the pattern attributes values and the corresponding minimum hypercube values:

$$F(S) = F(\sum_{j=1}^{d} w_j^c \left| x_j^c - hmin_j^c \right|), \tag{1}$$

where F is the activation function. The activation function could be a simple linear function, a piecewise linear function, a sigmoid or an adaptive function. The $\left| x_j^c - hmin_j^c \right|$ terms in effect provide the coordinates of the pattern within the hypercube, taking the lower left vertex of the hypercube as the origin: in other words, the vector of the pattern within the hypercube space.

The weights associated with the neurons for those hypercubes that contain the pattern are updated using the formula:

$$w_{j,new}^c = w_{j,old}^c + \lambda e \left| x_j^c - hmin_j^c \right|, j = 1, .., d, \tag{2}$$

where λ is the learning rate for the weights set to a small number (e.g. 0.01) and e is the classification error. The classification error is determined as follows. If a pattern belongs to more than one hypercube then we check whether the pattern's class corresponds to the hypercube class and assign the classification

error $e = 1 - F(S)$, otherwise $e = -F(S)$. If a pattern belongs to only one hypercube and the patterns class corresponds to the class of the hypercube, then the error $e = 0$.

To calculate the performance of the method we use an approach that is similar to the way classification error is determined during the training process. If a pattern belongs to only one hypercube and the pattern's class coincides with the hypercube class then the classification error is 0, otherwise 1. In the case of the training set, when a pattern belongs to only one hypercube the classification error is 0. If the pattern belongs to more than one hypercube then the hypercube with the highest activation determines the membership of the pattern and subsequently the classification error, i.e. 0 if the class of the pattern and the hypercube's class are the same, 1 otherwise. If the pattern does not belong to any of the hypercubes then the highest number of dimensions within range determines the membership. When there is more than one hypercube with that maximum number of dimensions then the highest neuron activation is taken into account to define the membership and the classification error. Finally, if all of the pattern's dimensions are outside the range of the dimensions of all hypercubes then the classification error is set to 1. The performance is calculated as the percentage of the correctly classified patterns.

2.3 Hypercube with adaptive activation functions

In this section we describe the algorithm when the associated activation functions with each neuron/hypercube are adaptive piecewise linear functions. The algorithm combines the simultaneous adaptation of both the weights and the shape of the individual neuron activation functions as in the case of the Adaptive Function Neural Network introduced in [3], [2]. This is a type of modal learning [11] which involves the strategic combination of modes of adaptation and learning within a single artificial neural network structure. Two or more modes may proceed in parallel in different parts of the neural computing structure (layers and neurons), or they occupy the same part of the structure, and there is a mechanism for allowing the neural network to switch between modes.

The algorithm is the same the one described in subsection B but the activation functions and the formula for updating the weights are different. Moreover, the activation functions are adapted during training. The weight components are initialised the same way as in B, but in addition they are normalised, in order to provide a stable range over which the functions can be adapted. We calculate the range of the intervals for the piecewise linear activation functions using the minimum and maximum value for the sums from formula (1) for all hypecrubes and training patterns. Then the activation functions values for each neuron are initialised to random numbers between 0 and 1 with function points evenly spaced across the calculated range. The number of points (intervals) in the activation piecewise functions is set according to the required precision. This is based on the smallest interval that is significant in the data set. During the training we calculate $F(S)$ using (1), and find the two neighbouring points that bound $F(S)$, which we denote with P_{i-1} and P_i.

We denote the values of the activation function at the points P_{i-1} and P_i as $Fval_i$ and $Fval_{i-1}$ respectively, and the slope of the activation function between these two points as $Fslope_i$. These values will be adapted in proportion to their proximity to $F(S)$. Following the algorithm from subsection B we update the weights and the activation functions in parallel for the hypercubes that contain a training pattern using:

$$w^c_{j,new} = w^c_{j,old} + \lambda e Fslope_i \left| x^c_j - hmin^c_j \right|, j = 1, \ldots, d$$

$$Fval_i^{new} = Fval_i + \mu eq_i$$

$$Fval_{i-1}^{new} = Fval_{i-1} + \mu eq_{i-1},$$

where the notations for the weights update are the same as in subsection B, $Fslope_i$ is the slope of the activation function at the particular interval, μ is the learning rate for the activation functions set to a small number (e.g. 0.1). The quantities q_i and q_{i-1} are the proximal-proportional values used to apportion the function adaptation across the two function points that bound the weighted sum input to the function, according to their proximity to the weighted sum input:

$$q_i = \frac{P_i - S}{P_i - P_{i-1}}$$

$$q_{i-1} = \frac{S - P_{i-1}}{P_i - P_{i-1}}$$

The classification error e is calculated the same way as in subsection B. Fig. 2 illustrates an adaptive linear piecewise function.

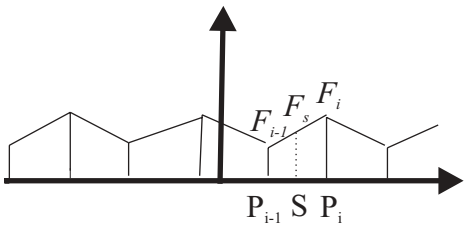

Fig. 2. Adaptive linear piecewise activation function

3 Experiments and Results

A range of data sets are chosen to represent a variety of learning challenges. They vary in terms of the number of input variables, the number of classes, and the level of separability of the classes. Since they are all known and freely available they provide useful benchmark comparisons with a number of neural computing and other machine learning techniques.

3.1 Data Sets

Animal data The Animal data presents a simple classification problem. It is artificial data and consists of 16 animals described by 13 attributes such as size, number of legs etc. [12]. The 16 animals are grouped into three classes (the first one represents bird, the second represents carnivore and the third represents herbivore).

Breast Cancer Wisconsin This data set was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [13], [14]. The data set used in the experiments consists of 683 patterns, 9 attributes and 2 classes with distribution 65.5% and 34.5%.

Ecoli The Ecoli data set contains 336 patterns with 7 attributes and 8 classes, which are the 'localization sites' [15]. 91% of the patterns belong to 4 classes and the rest to the remaining 4 classes.

Iris data The Iris data set has three classes setosa, virsicolor and virginica [16], [17]. The iris data has 150 patterns, each with 4 attributes. The class distribution is 33.3% for each of 3 classes. One of the classes is linearly separable from the other two, and the two are linearly inseparable from each other.

Optical and pen-based recognition of handwritten digits (OCR) data The OCR data set [18], [19] consists of 3823 training and 1797 testing patterns. Each pattern has 64 attributes which are integer numbers between 0 and 16. There are 10 classes corresponding to the digits 0 to 9. The 64 attributes are extracted from normalised bitmaps of handwritten digits by 43 people.

The experiments with the OCR data set use the already existing division of training and testing Patterns, 3828 and 1797 respectively, as originally proposed by Kaynak [19]. This facilitates direct performance comparisons with alternative algorithms that have been applied to the same data.

Pima Indians Diabetes The data set originates from the National Institute of Diabetes and Digestive and Kidney Diseases donated in 1990 by V. Sigillito, The Johns Hopkins University. It has 768 patterns, 8 numeric attributes and 2 classes with distribution 65% and 35%.

Tic-Tac-Toe Endgame This data set encodes the complete set of possible board configurations at the end of tic-tac-toe games, where "x" is assumed to have played first [20]. The target concept is "win for x" (i.e., true when "x" has one of 8 possible ways to create a "three-in-a-row"). There are 958 patterns, 9 attributes and 2 classes with distribution 65.3% and 34.7%.

Yeast This data set [15] consists of 1484 patterns, 7 attributes and 10 classes with most of patterns belonging to 4 classes.

Wine data The Wine data set is the result of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars [21]. The analysis determines the quantities of 13 constituents (input variables) found in each of the three types of wines. There are 178 patterns with the following distribution: class 1 (59 patterns), class 2 (71 patterns), class 3 (48 patterns).

Zoo data This is a simple data set with 101 patterns, 16 attributes and 7 classes.

3.2 Experiments

In the experiments, 20% selections of the patterns of each data set are allocated for testing and the remaining 80% form the training set. For each run the training and testing patterns are selected at random from the entire data set. The results are based on 10 runs. The Hypercube algorithm with a simple neuron requires the learning rate for the weight learning to be set as well as the number of epochs. The experiments for the data sets under investigation show that the algorithm is not sensitive to the learning rate which is set to a small number between 0.01 and 0.1. The number of epochs required for convergence is between 100 and 200. For the Hypercube algorithm with adaptive function the number of points for the piecewise functions has to be set, as well as the learning rates for the weights and the adaptive functions. The experiments reveal that 500 epochs are sufficient for training in all cases and the learning rates have to be small numbers between 0.01 and 0.1. The algorithm is most sensitive on the number of function points which varies between 10 and 50 for the data sets under consideration.

3.3 Results

Table 1 presents the results from the experiments. The notations in the Table 1 are as follows: HC is the hypercube algorithm without neurons, HCN is the hypercube algorithm with a simple neuron and HCAF is the hypercube algorithm with adaptive function. Table 2 presents comparative results for 7 of the data sets based on the results with three other algorithms: the classic Nearest Neighbour Classifier (1NN), the Batch Nested Generalised Exemplar(BNGE) and the Evolutionary Selection by CHC (EHS-CHC)[8]. The BNGE is a batch version of the first NGE model and it addresses some of the limitations of NGE [4]. We compare with these methods because BNGE appears to yield the best results for the 7 data sets in Table 2, while the EHS-CHC gives the best results average results over all data sets as reported in [8]. The HCAF average results are better than the average results of the BNGE and EHS-CHC over the selected 7 data sets. HCAF gives better results in 4 of the data sets compared to BNGE and is very close in 2 others. HCAF performs better in 6 data sets in comparison to EHS-CHC.

The results for the Breast cancer and OCR data sets are comparable to the results obtained with other classifiers. For example, 95.9% (96.5% for HCAF) accuracy is reported for the Breast cancer data set in [22] using a multisurface method of pattern separation and linear programming. Using combination of four learning modes, Snap-Drift and Adaptive Functions in [2], 94.99% (92% for HCAF) accuracy is obtained for the OCR data set.

4 Conclusions

Hypercube is computationally very efficient, especially in comparison to NGE methods. Both in terms of computation and classification, the results achieved

Table 1. Mean % correct classification for train and test sets based on 10 runs. Standard deviation is given in italic below the mean %.

Method	HC train	HC test	HCN train	HCN test	HCAF train	HCAF test
Animal	100	92	100	100	100	100
st. dev.	*0*	*11.2*	*0*	*0*	*0*	*0*
Breast	90.3	90.1	93.9	92.3	98.2	96.5
st. dev	*1.1*	*2.2*	*2.0*	*2.7*	*0.6*	*1.7*
Ecoli	78.5	72.9	92	82.4	99.6	84.4
st. dev.	*1.3*	*5.1*	*0.9*	*2.6*	*0*	*2.7*
Iris	95.1	94.4	98.9	94.6	98.8	95.8
st. dev.	*1.9*	*2.9*	*0.8*	*1.2*	*1.3*	*1.4*
OCR	36	36	97	91	99	92
st. dev.	*0*	*0*	*0*	*0*	*0*	*0*
Pima	67.6	66.4	67.5	65.5	77.1	76.6
st. dev.	*1.3*	*1.8*	*1.6*	*2.5*	*0.7*	*3.1*
Tic-tac	65.5	36.9	98.7	98.2	98.7	98.2
st. dev.	*1.1*	*3.7*	*0.4*	*0.9*	*0.4*	*0.6*
Wine	94.7	93.1	100	96.7	100	98.3
st. dev.	*0.9*	*3.2*	*0*	*2.2*	*0*	*2.2*
Yeast	35.2	34.8	54.3	51.3	57.4	55.9
st. dev.	*0.9*	*2.8*	*1.5*	*2.5*	*1.1*	*1.5*
Zoo	96.3	75	98.3	94	100	96.5
st. dev.	*5.7*	*5.4*	*0.4*	*5.1*	*0*	*2.4*

Table 2. Mean % correct classification test sets - comparison with classic and best hyperrectangle methods

Method	1NN	HCAF	BNGE	EHS-CHC
Ecoli	80.7	84.4	82.16	81.54
Iris	93.3	95.8	96	94.0
Pima	70.33	76.6	72.78	75.01
Tic-tac	73.07	98.2	92.07	92.06
Wine	95.52	98.3	96.6	94.31
Yeast	50.47	55.9	57.35	56.34
Zoo	92.81	96.5	96.83	95.00
Average	*79.45*	*86.53*	*84.83*	*84.04*

demonstrate the advantages of a simplified hypercube method that is treated as a mode of learning. Rather than perform the computationally intensive process of optimising hyperrectangles, we achieve high levels of classification by combining hypercubes with other modes. Whilst these modes carry a computational overhead, they are only invoked for the minority of patterns that reside in the overlapping regions of hypercubes. Each class is characterised by only one hypercube and a single neuron, and so the computation required scales linearly with the number of classes. The combination of the hypercube and neural modes of learning proves effective on the range of data sets considered in this work. Further investigation will focus on combining hypercubes with alternative unsupervised and supervised forms of modal learning, and applying the techniques to wider range of problems including large scale data sets. A receiver operating characteristic (ROC) analysis will be also carried out to evaluate the prosed algorithms in terms of true positive, false positive, false negative and tru engative values.

References

1. Salzberg, S.: A nearest hyperrectangle learning method. Machine Learning. 6, 251–276 (1991)
2. Kang, M., Palmer-Brown, D.: A Modal Learning Adaptive Function Neural Network Applied to Handwritten Digit Recognition. Information Sciences. 178. 20, 3802–3812 (2008)
3. Kang, M., Palmer-Brown, D.: A Multilayer Adaptive Function Neural Network (MADFUNN) for Analytical Function Recognition. In IJCNN (2006) part of the IEEE World Congress on Computational Intelligence, WCCI 2006, Vancouver, BC, Canada, pp. 1784-1789. (2006)
4. Wettschereck D., Dietterich, T.G.: An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. Machine Learning. 19. 1, 5–27 (1995)
5. Carpenter, G. A., Grossberg, S., Markuzon, N., Reynolds, J. H., Rosen, D. B.: Fuzzv ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. IEEE Transactions on Neural Networks. 3. 5, 698–712 (1992)
6. Wilson, D. R., Martinez, T. R.: Reduction Techniques for Instance-Based Learning Algorithms. Machine Learning. 38. 3, 257–286 (2000)
7. Garcia, S., Cano, J.R., Herreraa, F.: A memetic algorithm for evolutionary prototype selection: A scaling up approach. Pattern Recognition. 41. 8, 2693–2709 (2008)
8. Garcia, S., Derracb, J., Luengob, J., Carmonab, C., Herreraa, F.: Evolutionary Selection of Hyperrectangles in Nested Generalized Exemplar Learning. Applied Soft Computing. doi:10.1016/j.asoc.2010.11.030 (2010)
9. Simpson, P.K.: Fuzzy min-max neural networks. I. Classification. IEEE Transactions on Neural Networks. 3. 5, 776–786, DOI 10.1109/72.159066 (1992)
10. Furmanski, G. C. F.: Hypercube Algorithms for Neural Network Simulation The Crystal_Accumulator and the Crystal_Router. In: Proceedings of the third conference on Hypercube concurent computers and applications: Architecture, software, computer systems, and general issues. 1, pp. 714-724. (1988)

11. Palmer-Brown, D., Lee, S. W., Draganova, C., Kang, M.: Modal Learning Neural Networks. In WSEAS Transactions on Computers. 8. 2, 222–236 (2009)
12. Ritter, H., Kohonen, T.: Self-Organizing Semantic Maps. Biological Cybernetics. 61, 241–254 (1989)
13. Mangasarian, O. L., Wolberg, W. H.: Cancer diagnosis via linear programming. SIAM News. 23. 5, 1–18 (1990)
14. William, H., Wolberg, Mangasarian, O.L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In: Proceedings of the National Academy of Sciences, U.S.A., vol. 87, pp. 9193-9196 (1990)
15. Horton, P., Nakai, K.: A Probablistic Classification System for Predicting the Cellular Localization Sites of Proteins. Intelligent Systems in Molecular Biology, 109–115 (1996)
16. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annual Eugenics. 7. Part II, pp. 179-188, 1936; also in Contributions to Mathematical Statistics. John Wiley, NY (1950)
17. Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis, pp. 218. John Wiley & Sons, New York (1973)
18. Alpaydin, E., Kaynak, C.: Cascading Classifiers. Kybernetika. 34, 369–374 (1998)
19. Kaynak, C.: Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition. MSc Thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University (1995)
20. Aha, D. W.: Incremental constructive induction: An instance-based approach. In: Proceedings of the Eighth International Workshop on Machine Learning, pp. 117-121 (1991)
21. Forina, M., Lanteri, S., Armanino, C. et al.: PARVUS–an extendible package for data exploration, classification and correlation. Institute of Pharmaceutical and Food Analysis and Technologies, Via Brigata Salerno, 16147 Genoa, Italy (1991)
22. Wolberg,W. H., Mangasarian, O. L.: Multisurface method of pattern separation for medical diagnosis applied to breast cytology. In: Proceedings of the National Academy of Sciences, vol. 87, pp. 9193-9196 (1990)