UNIVERSITY OF CINCINNATI

Date: September 6, 2006

I, Danny W. Vance,

hereby submit this work as part of the requirements for the degree of:

Doctor of Philosophy

in:

Computer Science and Engineering

It is entitled:

An All-Attributes Approach to Supervised Learning

This work and its defense approved by:

Chair: Dr. Anca L. Ralescu

Dr. Michael Behbehani

Dr. Kenneth Berman

Dr. Raj Bhatnagar

Dr. Urmila Ghia

An All-Attributes Approach to Supervised Learning

A dissertation submitted to the

Division of Research and Advanced Studies

of the

University of Cincinnati

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in the

Department of Electrical and Computer Engineering

and Computer Science

of the College of Engineering

by

Danny W. Vance

B.S. (College of Education), August 1978

University of Cincinnati, Cincinnati, U.S.A.

M.S. (College of Arts & Sciences), August 1983

University of Cincinnati, Cincinnati, U.S.A.

M.S. (College of Engineering), August 1987

University of Cincinnati, Cincinnati, U.S.A.

Advisor & Committee Chair: Prof. Anca L. Ralescu

September, 2006

Abstract

The objective of supervised learning is to estimate unknowns based on labeled training samples. For example, one may have aerial spectrographic readings for a large field planted in corn. Based on spectrographic observation, one would like to determine whether the plants in part of the field are *weeds* or *corn*. Since the unknown to be estimated is categorical or discrete, the problem is one of classification. If the unknown to be estimated is continuous, the problem is one of regression or numerical estimation. For example, one may have samples of ozone levels from certain points in the atmosphere. Based on those samples, one would like to estimate the ozone level at other points in the atmosphere.

Algorithms for supervised learning are useful tools in many areas of agriculture, medicine, and engineering, including estimation of proper levels of nutrients for cows, prediction of malignant cancer, document analysis, and speech recognition. A few general references on supervised learning include [1], [2], [3], and [4]. Two recent reviews of the supervised learning literature are [5] and [6]. In general, univariate learning tree algorithms have been particularly successful in classification problems, but they can suffer from several fundamental difficulties, e.g., *"a representational limitation of univariate decision trees: the orthogonal splits to the feature's axis of the sample space that univariate tree rely on"* [8] and *overfit* [17].

In this thesis, we present a classification procedure for supervised classification that consists of a new univariate decision tree algorithm (Margin Algorithm) and two other related algorithms (Hyperplane and Box Algorithms). The full algorithm overcomes all of the usual limitations of univariate decision trees and is called the Paired Planes Classification Procedure. The Paired Planes Classification Procedure is compared to Support Vector Machines, K-Nearest Neighbors, and decision trees. The Hyperplane Algorithm allows direct user input as to acceptable error for each class as contrasted with indirect input (through use of a *slack variable*) with Support Vector Machines. Theoretical and real-life datasets results are shown. Experiments on real-life datasets show that error rates are in some circumstances lower than these supervised learning algorithms, while usually being computationally less expensive by an order of magnitude (or more).

Dedicated to my family.

Acknowledgements

A dissertation marks a milestone in the intellectual achievement of an individual. In fact, it is much more. It is an entry point into the world inhabited by others who in the highest degree value knowledge and exploration of ideas. It is the culmination of the efforts of many people, of whom the individual writing the dissertation is but one. The acknowledgements here try to reflect this.

My mother deserves the most credit. It was she who was my first teacher and always stood behind me in whatever I did. Thank you mom, wherever you are. Anca Ralescu - my teacher, then advisor and friend – I cannot thank you enough. Without your belief in me, I simply would never have finished. Most of the other members of the dissertation committee were also my teachers. It is through their teachings and criticisms that I became a better researcher. I appreciate the efforts of all members of my dissertation committee. Thanks to all of you for that most precious commodity that you gave – time.

My wife and children have had the patience, love, and understanding to go through the long process of my schooling and research. They wished me the best with my conference papers and welcomed me back from each trip. They are the icing on the cake. To all of you in my thread of life – thank you for the connections.

Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 THE SUPERVISED LEARNING PROBLEM

Supervised learning can be viewed as a method for function approximation from training data. The objective of supervised learning is to estimate unknowns based on labeled observations, input-output pairs. The input is typically a vector. If the output is a class label, then the problem is called classification. Classification places individual items into groups/classes based on quantitative information inherent in the items (variables, traits). If the output is a continuous variable, the problem is called regression.

For instance, one may have aerial spectrographic readings for a large field planted in corn. Based on spectrographic observation of a particular area of this field, one would like to determine whether the plants in this area of the field are *weeds* or *corn*. Since the unknown to be estimated is categorical or discrete, the problem is one of classification.

On the other hand, one may have samples of ozone levels from certain points in the atmosphere. Based on those samples, one would like to estimate the ozone level at other points in the atmosphere. Since the unknown to be estimated is continuous, the problem is one of regression.

Supervised learning algorithms are useful tools in many areas of automated recognition (handwriting, speech, image, chemical compound), agriculture (plant identification, automatic livestock nutrient and medication dosage), medicine (disease classification, deciding dosage level), and engineering (fault diagnosis, target identification, terrain characterization for automated driving). General references on supervised learning are readily available. Two recent reviews of the supervised learning

literature are [5] and [6]. Supervised learning may be used as an end goal. It also may be used as a preprocessing step for other work. For example, classification of data into disease or disease-free categories might precede a study of dosage level for the disease category.

1.2 SUPERVISED LEARNING ALGORITHMS

Assuming that the type of data has been decided and training examples have been collected, the input attributes must be decided. Often a subset of the attributes is used to avoid computational complexity and other difficulties, such as *the curse of dimensionality* (section *1.3.1*) and *overfit* (section *1.3.2*). The attributes may or may not be transformed by some process into a *feature*. The value for an attribute might be squared or two attributes multiplied together to form such a *feature*.

Properly speaking, *attribute* refers to the name of the variable and *feature* refers to the value of the variable. However, these are often used interchangeably. Additionally, *feature* often refers to a combination of two or more attributes, as described above. Within this study, we never use *feature* in this sense when referring to our classification procedure. However, it may be used when referring to other algorithms, particularly multivariate decision trees.

Next, the design of the learning function is decided. The design is tied to the decision as to use of a subset of attributes and whether to create *features*. Finally, using parameters determined from training the learning function, the results are tested on a dataset previously unseen by the algorithm.

We may divide supervised learning into global models and local models. Here, global means that all training examples are used to classify a new point and local means only nearby points are considered to classify a new point. The assumption for the local model is that the point to be classified is more similar to nearby points than those points far away.

1.2.1 *K*-NEAREST NEIGHBOR

The *k*-Nearest Neighbor algorithm [*k*-NN] is an example of a local model. It is a method of classification that relies on grouping examples that are close to one another in the feature space. The Euclidean distance is usually used as a proximity measure. If a weighted average of the *k* nearest neighbors is used, the effect of outliers (isolated examples) that are noisy (errors) is lessened.

The training phase of the algorithm stores the vectors. At the time of classification, distances from the point to be classified to all stored vectors are calculated. The *k*-nearest neighbors are then selected. The point is classified as to whichever class in this set of neighbors that is the most frequent; *k* odd ensures that there are no ties. When the distance is calculated, all attributes are used, whether relevant or not. This can have a detrimental effect on classification. This problem of many irrelevant attributes is known as *the curse of dimensionality* and is detailed in section 1.3.1.

There are methods, such as weighing each attribute differently, to try to overcome *the curse of dimensionality*. This has the effect of giving less value to the irrelevant attributes. Another method is to completely eliminate those attributes judged irrelevant. Much research has been applied to these two areas in order to improve classification.

The *k*-Nearest Neighbor algorithm is easy to implement. However, as the number of training examples increases, the computation becomes very costly. Thus, for large

datasets, efforts have centered on methods (partial distances, prestructuring, and editing the stored prototypes) to overcome this limitation [1].

The computational complexity during testing is a limitation. For n training samples with d attributes, if one seeks the single ($k = 1$) closest point to a test point $\mathbf{x}$, the Euclidean distance calculation is $O(d)$ and the search is $O(dn^2)$ [1], where d is the number of attributes. For $k > 1$, the k closest points vote as to which class the new point belongs to and that class is assigned. To do this, a sort is required on the n points to find the k closest points: $O(n \ln n)$. Therefore, the search complexity is $O(dn^2 + n \ln n)$.

1.2.2 SUPPORT VECTOR MACHINES

Support Vector Machines (SVMs) [11] are a special type of binary classifier that seeks to separate classes by a linear decision boundary – a hyperplane. The distance from the decision boundary to the nearest data points, support vectors, is called the *margin* (*of separation*). It uses only the hyperplanes determined by these vectors to classify new points and is therefore a global model conceptually and a local model for actual classification.

The *optimal hyperplane* is one where the margin is maximized (there is a maximum distance to the closest vectors from both classes), minimizing the number of support vectors. A hyperplane with a maximum margin allows more accurate classification of new points. A hard margin SVM is applicable when the classes are linearly separable. A soft margin SVM is applicable when the classes are non-separable, and the margin is chosen to minimize errors of classification.

SVMs are scale dependent and slow training: "*One disadvantage of SVM is that the training time scales somewhere between quadratic and cubic with respect to the*

number of training samples." [12]. The complexity is $O(n^3)$ according to [11], but even with a quadratic complexity this is still unmanageable for large n.

1.2.3 DECISION TREES

Decision Trees are an example of a global model. By contrast, to k-Nearest Neighbor algorithm and Support Vector Machines, decision trees such as ID3 (*interactive dicotomizer*) [7] and C4.5 [13] typically select only a subset of the attributes to form the hypothesis – a decision tree. ID3 was restricted to attributes with discrete values for not only the output variable, but also the input variables. C4.5, Quinlan's extension of ID3, allows continuous-valued attributes for the input variables as well as methods to prune the tree. However, according to Quinlan, *"Several authors have recently noted that C4.5's performance is weaker in domains with a preponderance of continuous attributes than for learning tasks that have mainly discrete attributes."* [14].

Both algorithms start by choosing as the root node of the tree the *best* attribute with respect to the information gain. For each of the possible discrete values of this attribute a descendant node is created and the relevant training examples assigned to the correct node. The process is repeated using the training examples at each node and so on.

This is a *greedy* approach and the ID3 algorithm, in its pure form, never goes back to reconsider: *"It does not have the ability to determine how many alternative decision trees are consistent with the available training data,"* [4]. Thus, it has an incomplete search of the hypothesis space of all possible decision trees. This is referred to as a *search* bias.

All the relevant training examples are considered at each step making it less sensitive to single examples. In classifying, the first acceptable tree is chosen.

This preference is essentially for a shorter tree [4]. When a preference is for the simplest hypothesis that fits the data, it is called *Occam's razor*. However, objections can be made to this. For instance, if we have two hypotheses, such as two decision trees, that are equally simple, how do we choose which one to use? Another objection is that the size of the hypothesis, determined by the internal representation of the learner, can be different for the *same* hypothesis by two different learners. Mitchell [4] gives an example where the same learned decision tree could be represented by two different learners, each justifiable by *Occam's razor*, which generalize differently. Yet another objection is that a more complex hypothesis can actually provide a better explanation for the classification, i.e., not everything has a simple explanation.

Choosing attributes, how deep a tree to grow a decision tree, and how to handle missing data are all addressed by C4.5. The depth of the tree is of particular importance. *Overfitting* can result if a tree is grown too deep. A more detailed explanation of *overfit* is in section 1.3.2.

CART [15] is a decision tree algorithm similar in most respects to C4.5, with the notable exception that its internal nodes test on linear combinations of attributes [16] [15]. "*The basic methodology of divide and conquer described in C4.5 is also used in CART. The differences are in the tree structure, the splitting criteria, the pruning method, and the way missing values are handled.*" [17].

Real-valued variables are treated the same way, multiway splits are used with nominal data, and heuristics based on statistical significance of splits are used for pruning the tree [1].

They handle missing attribute values differently: CART uses *surrogate splits* while C4.5 follows *all* possible answers to the leaf nodes. Surrogate splits may use the

simple measure of counting the number of instances sent to the *left* and *right* by each of the two possible splits and choose the one with a higher count. C4.5 considers the class labels of the leaf nodes reached and weighs its decision by the probability at the splitting node of how any instance would be classified.

We now discuss the computational complexity for CART and C4.5. Each is given in terms of the two-class problem.

Duda gives the training complexity for CART as $O(kn^2 \log n)$ and classification complexity as $O(\log n)$, where k is the number of attributes and n is the number of training points [1]. Implicit in these calculations are two assumptions: i) an average case that splits the data into halves for each branch of the binary tree at every level and ii) there is a single training point per leaf node.

The training complexity for C4.5 is of order $O(kn \log n) + O(n (\log n)^2))$ [18], as shown below:

- Assume k attributes, n training instances and a tree depth of $O(\log n)$
- Cost for building a tree: $O(kn \log n)$
- Complexity of subtree replacement: $O(n)$
- Complexity of subtree raising: $O(n (\log n)^2)$
- Every instance may have to be redistributed at every node between its leaf and the root: $O(n \log n)$
- Cost for redistribution (on average): $O(\log n)$
- Total cost: $O(kn \log n) + O(n (\log n)^2)$

Univariate decision trees are trees that test one attribute at a time. They alleviate *the curse of dimensionality* to some degree, but typically have several limitations [8]. These are listed below.

1. Trees (such as C4.5 [13] or CART [15]) usually test the same attributes in one or more subtrees. For example, consider the *Digit Recognition* (our label) decision tree from page 47 of [15] shown in Figure 1-1 (originally labeled as FIGURE 2.13 in [15]). The attributes in this example correspond to the lights displayed to form digits on electronic watches and calculators. By [15], the device is faulty and the problem is to decide which digit is displayed. The root of the tree, t_1, is the attribute x_5. The subtrees to the left and right each contain the attributes x_2, x_3, and x_4. The attribute x_4 is used multiple times to determine the class/digit:

   $(x_5 = 0 \wedge x_4 = 0 \wedge x_1 = 0)$ leads to class *1*.

   $(x_5 \neq 0 \wedge x_2 \neq 0 \wedge x_4 = 0)$ leads to class *10*.

2. An attribute may occur more than once in a path. For a continuous attribute, such as those used in the *Waveform Recognition* (our label) decision tree from page 54 of [15] shown in Figure 1-2 (originally labeled as FIGURE 2.13 in [15]), the attribute may occur more than once corresponding to the attribute value interval being split more than once. For example, x_6 is split into two intervals: $x_6 \leq 2.0$ and $x_6 > 2.0$. The left-most side ($x_6 \leq 2.0$) is then split again into two intervals: $x_6 \leq 0.8$ and $x_6 > 0.8$. The root of the tree is the attribute x_6. The subtree to the left contains the attribute x_6 again as well and the left-most path to class *3* is $(x_6 \leq 2.0 \wedge x_{11} \leq 2.5 \wedge x_6 \leq 0.8)$. Multiple paths to class *3* lead to a disjunction of conjunctions:

$$(x_6 \leq 2.0 \wedge x_{11} \leq 2.5 \wedge x_6 \leq 0.8) \vee$$

$$(x_6 \leq 2.0 \wedge x_{11} > 2.5 \wedge x_{15} > 1.9) \vee$$

$$(x_6 > 2.0 \wedge x_{10} > 2.6 \wedge x_7 \leq 0.9)$$

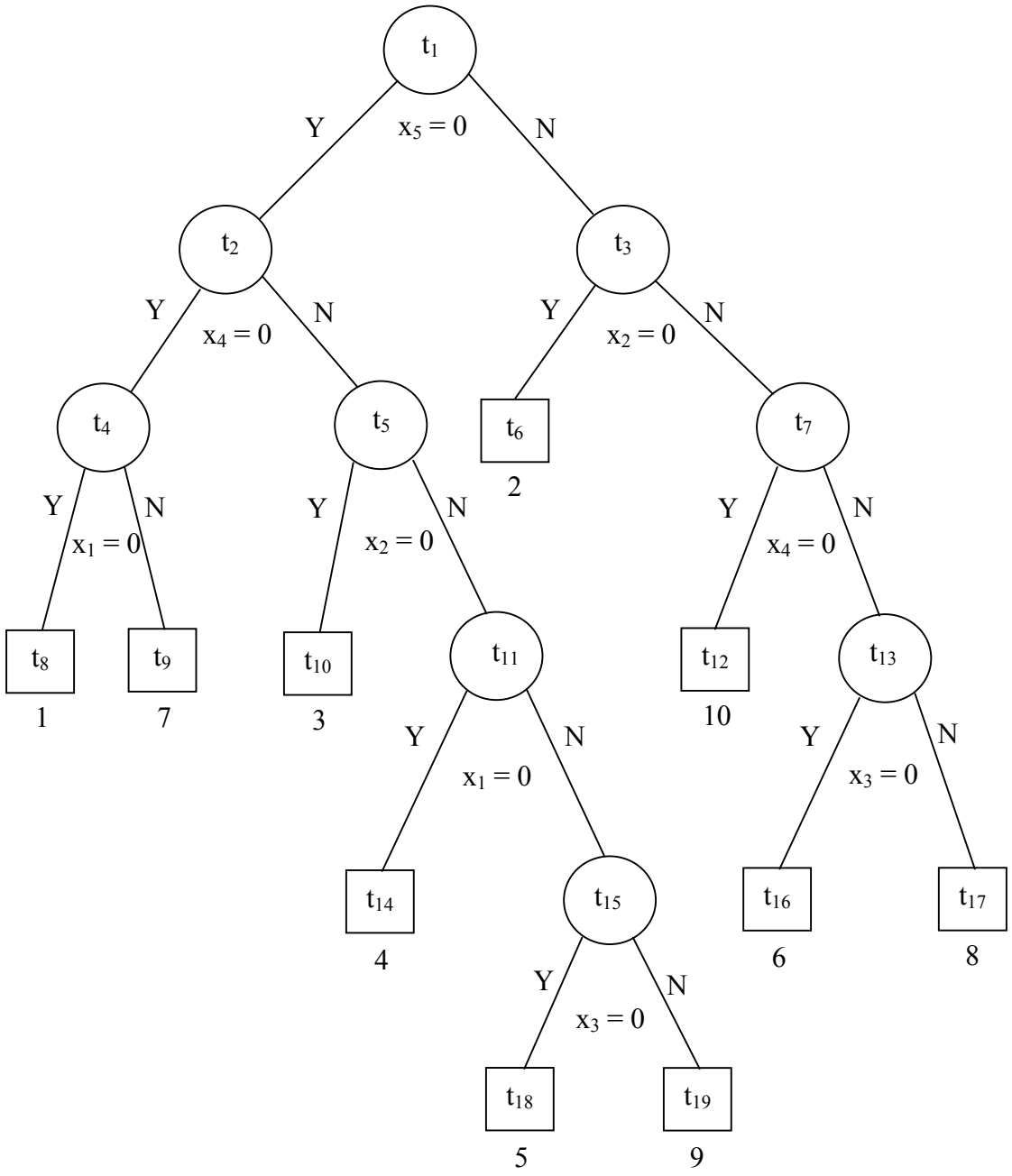


Figure 1-1. *Digit Recognition* Decision Tree [15].

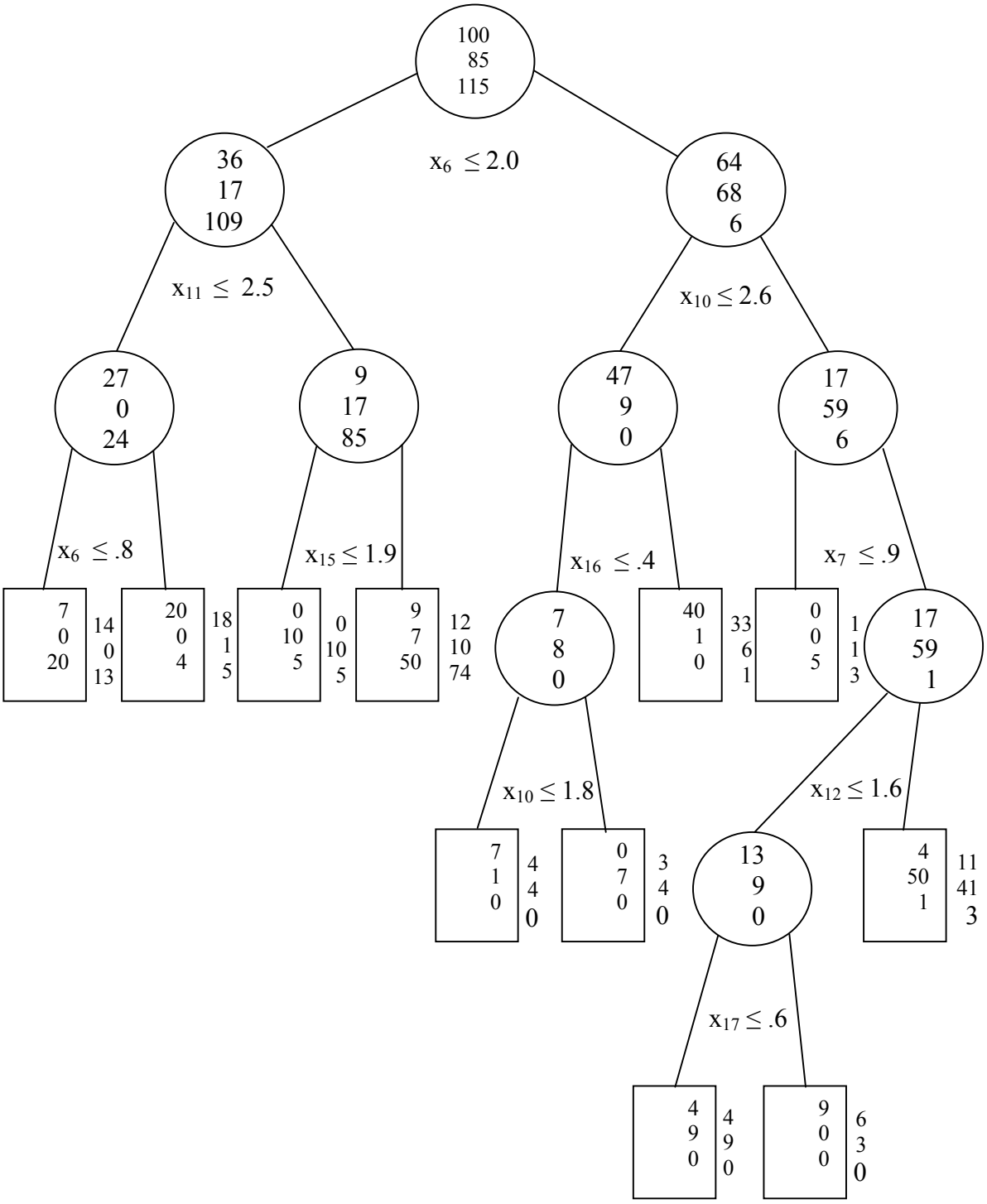


Figure 1-2. *Waveform Recognition* Decision Tree [15].

3. Univariate binary trees rely on splits orthogonal to the axes of the feature space. In a multi-dimensional feature space, this may be too constrained a model to represent accurately the decision boundaries, i.e., an oblique decision boundary may split the classes better.

1.3 ISSUES IN SUPERVISED LEARNING ALGORITHMS

We now consider the three algorithms presented in section 1.2 and discuss some of these shortcomings relative to them. Many of these problems are common to all supervised learning and efforts to ameliorate them are important. Complexity of computation has already been mentioned for SVMs and the *k*-Nearest Neighbor algorithm. We will look at other issues here.

1.3.1 THE CURSE OF DIMENSIONALITY

Imagine a case where there are 100 attributes, but only one of these is relevant in the classification. For the *k*-Nearest Neighbor algorithm, when the distance is calculated, all attributes are used. This can have a detrimental effect. All 100 attributes are still used to calculate the distances. There may be points that have the same value for the relevant attribute but are far away from each other in the 100 dimensional space. Thus, the non-relevant attributes dominate the distance measure. The effect of irrelevant attributes is felt in other algorithms as well. This problem of many irrelevant attributes is known as the *curse of dimensionality* [a term coined by Richard Bellman].

1.3.2 OVERFITTING

Overfitting occurs when improvement of classification accuracy on training is done at the expense of overall accuracy (including testing). Consider the hypothesis *h* a tree (obtained by a decision tree algorithm) represents. It has certain accuracy on the training

data with hypothesis *h*. Now let there be *noise* introduced in one of the examples, such as a mislabeling of its class and let *h′* be the hypothesis corresponding to the tree obtained by the re-training. This tree separates this example from other examples and, with this example, is a more complex tree. The resulting hypothesis *h′* fits the training data better than *h*, but in fact the true accuracy for the training data is reduced. The new hypothesis has learned the error. Even when the data is noise-free, *overfitting* can occur. If there are only a few examples at a leaf node, coincidence can cause misclassification. Essentially this is the result of a too-small sample size affecting the statistical test, which relies on a minimum number of examples.

Decision trees, including C4.5, are generally prone to overfitting: "... *if there are no conflicting cases, the decision tree will correctly classify all training cases. This so-called overfitting is generally thought to lead to a loss of predictive accuracy in most applications (Quinlan 1986).*" [17].

1.3.3 STRUCTURAL REPRESENTATION LIMITS

In the case of Support Vector Machines, a hyperplane is used, but a curved surface may actually better separate the classes. The *kernel trick* is a method of dealing with this. Rather than using a linear surface, the dot product is replaced by a non-linear function of the dot product, which corresponds to the dot product in the higher dimensional feature space, hence the name *kernel*. The classifier is linear in a higher dimensional space but non-linear in the original input space.

Univariate decision trees use decision surfaces that are orthogonal to the axes. Unfortunately, a consequence of this is that classes separable by oblique lines may not be separated as well by univariate decision trees. Multivariate decision trees are trees that test more than one attribute at a time. They attempt to address this disadvantage of

univariate trees by being able to create an oblique split. This is done by a linear combination of two or more features. There is an inherent difficulty in this method: Which features does one choose as a subset? To date, results have not been impressive. The multivariate decision tree algorithms also are of increased computational complexity.

One approach (*C-Net*) to creating multivariate decision trees is to use a neural network in combination with Quinlan's C5 (an improvement to C4.5) and thereby create the tree [19]. The artificial neural network (ANN) is trained on the data, then the output of the hidden layer is the input feature vector to C5. Finally, the univariate decision tree in the new feature space of hidden units is converted to a multivariate decision tree.

Figure 1-3 (reproduced from [19] and originally labeled as *Table 2*) shows results for testing on four real-life datasets and on four artificial datasets. As can be seen when compared to the ANN, there is minimal improvement in error rates for the real-life datasets and an increase in the error rates for the artificial datasets. Comparison to C5 is more favorable, with the Liver dataset showing the greatest improvement. However, all these error rates for C-Net appear to us to be strongly linked to the error rates for the ANN. This is at a cost of computational complexity.

Problem	C-Net	C5	ANN
Breast Cancer	2.2% ± 1.6†	4.9% ± 2.3	2.5% ± 1.8
	2.0 ± 0.5	20.3 ± 1.0	2
Haberman	28.4% ± 2.3	28.8% ± 4.6	28.8% ± 4.6
	3.3 ± 0.6	4.0 ± 1.6	2
Liver	31.7% ± 7.8‡	37.4% ± 5.5	32.6% ± 7.9
	7.5 ± 3.0	21.5 ± 5.7	5
Dairy	22.7% ± 0.4	22.9% ± 0.4	23.3% ± 1.4
	2.0 ± 0.0	8.4 ± 3.5	2
P2	2.3% ± 1.2	2.7% ± 1.1	1.4% ± 1.2
	30.3 ± 6.7	53.8 ± 4.0	4
P3	1.7% ± 0.4Φ	2.4% ± 0.8	0.5% ± 0.6
	31.1 ± 5.0	40.2 ± 1.4	5
P4	1.4% ± 0.8‡	2.6% ± 0.6	0.6% ± 0.7
	30.6 ± 2.0	47.6 ± 3.5	5
P5	2.6% ± 1.2Φ	3.4% ± 1.0	0.8% ± 0.6
	34.1 ± 12.6	40.0 ± 2.2	5

Figure 1-3. *C-Net* Multivariate Decision Tree [19].

In [20], an *omnivariate* decision tree is proposed. The decision node is allowed to be univariate, linear (multivariate), or nonlinear. C4.5 is used to construct the univariate tree and a single-layer perceptron is used at each node to construct the multivariate tree.

Table 1-1 (partially reproduced from [20] and originally labeled as *Table III*) shows partial results for testing on 30 real-life datasets from the UCI depository [21]. We restrict our comparison of classification accuracy to univariate versus multivariate trees (as created in [20]). Of the 30 sets, 18 are classified with higher accuracy by multivariate decision trees and 12 with higher accuracy by univariate decision trees. There is no basis for knowing which will perform better on an arbitrary new dataset, i.e., "*No single tree algorithm dominates or is dominated by others.*" [1]

Table 1-1. Linear vs. Multivariate Decision Trees [20].

Set	Uni	Lin	Mul
BAL	65.5,3.0	91.6,	2.0
BRE	94.6,1.8	96.2,	0.7
BUP	62.8,3.3	64.1,	3.8
CAR	86.0,1.6	92.7,	1.7
CMC	52.6,3.1	44.6,	2.2
CRE	84.7,1.0	82.5,	2.2
CYL	67.2,4.6	69.2,	2.7
DER	92.5,2.4	97.2,	1.0
ECO	78.2,4.0	80.0,	4.1
FLA	88.3,2.5	88.0,	2.6
GLA	60.0,5.5	57.3,	7.4
HAB	71.9,3.6	73.3,	3.2
HEP	78.9,4.4	81.5,	4.4
HOR	73.8,6.7	79.8,	6.6
IRI	92.9,3.3	95.4,	2.3
IRO	86.1,3.7	87.8,	2.8
MON	89.8,7.7	72.3,	5.6
MUS	99.8,0.1	99.9,	0.0
NUR	94.4,0.4	93.4,	0.6
OCR	84.8,0.8	92.4,	0.8
PEN	92.5,0.6	94.9,	2.7
PIM	70.7,2.9	73.9,	1.3
SEG	92.0,0.9	87.5,	10.6
SPA	91.4,0.7	91.1,	0.6
TIC	78.7,1.8	97.9,	0.7
VOT	95.6,0.6	95.4,	0.9
WAV	75.9,0.6	82.8,	0.9
WIN	86.6,1.9	96.0,	1.9
YEA	54.4,3.0	47.7,	4.4
ZOO	82.9,7.3	79.2,	9.5

1.4 PARALLEL PLANES CLASSIFICATION PROCEDURE (PPCP) – MAIN FEATURES

This dissertation presents a new, nonparametric method, the Paired Planes Classification Procedure (PPCP), for supervised learning that uses all attributes of a dataset as needed. PPCP can use continuous and discrete numbers for variables, whether based on a metric or numbered list. Many algorithms use only a subset of the attributes. This is done for both reduced complexity and to avoid *overfitting* from irrelevant attributes. Note that such attributes are discarded *forever*. However, "*Noise reduction and consequently better class separation may be obtained by adding variables that are presumably redundant.*" [22].

Additionally, for difficult class boundaries, such as nonlinear boundaries, other algorithms map into other feature spaces. While they can yield good performance, strictly speaking this is a different problem because they do not consider only the original feature space.

Our objective is to investigate to what extent classification in the original feature space is possible in terms of acceptable accuracy and computational complexity. The decision surfaces are in the original attribute space. To date, nine papers have been accepted at conferences, both domestic and international: [23], [24], [25], [26], [27], [28], [29], [30], [31].

Paired Planes Classification Procedure is a classification procedure based on component classifiers: the Hyperplane Algorithm, the Margin Algorithm, and the Box Algorithm. The component classifiers are based on the same underlying common approach of parallel hyperplanes, but differ in the way these parallel hyperplanes are

obtained. Each classifier has expertise in a particular region of the feature space. *"Such full classifiers are called mixture-of-expert models, ensemble classifiers, modular classifiers, or occasionally pooled classifiers. Such classifiers are particularly useful if each of its component classifiers is highly trained (i.e., an "expert") in a different region of the feature space."* [1].

For a two-class problem, classifiers typically divide the feature space into two regions - Class *1* and Class *2*. The work presented here typically divides the space into *three* regions - Class *1*, Class *2*, and unclassified, where the last is a region in which the classifier abstains, usually a region of overlap between Class *1* and Class *2*.

This produces trinary decision trees. It avoids the typical limitations of univariate decision trees that were detailed in section *1.2.3* and are listed again, in less detail, below.

1. Trees usually test the same attributes in one or more subtrees.
2. An attribute may occur more than once in a path.
3. Univariate binary trees rely on splits orthogonal to the axes of the feature space.

In looking at the region of overlap, we consider three cases. Our use of parallel planes covers all three cases. Because we use three (related) methods, PPCP can classify a wide range of datasets.

Case 1: no or little overlap

Many algorithms can accurately classify when there is no or little overlap. Support vector machines do this well, but there is a high computational cost.

For the case of no or little overlap, our method (the Hyperplane Algorithm) relies on hyperplanes to split the feature space. It finds a pair of hyperplanes in a straightforward, simple manner. Unlike Support Vector Machines, it uses two

hyperplanes and it does not rely on a set of support vectors. The amount of error acceptable for each class can be input directly and explicitly by the user (unlike SVMs).

The upper bound of complexity for the Hyperplane Algorithm is $O(kn)$, where k is the number of attributes and n is the number of samples. The complexity of our method is much less than that of SVMs: $O(n^3)$ [11].

Case 2: moderate overlap

For the case of moderate overlap, the Margin Algorithm relies on a univariate decision tree. Because only one or a few attributes are used to classify a point, univariate decision trees alleviate *the curse of dimensionality* to some degree. Our method uses only one attribute at a time, thus it has an advantage over most univariate trees. All the training examples are considered at each step to decide how to proceed. This makes it less sensitive to errors caused by single examples.

Univariate decision trees do typically have several limitations. Our classification procedure eliminates one of these in the first step, the Hyperplane Algorithm. Presumably, if the classes were separated better by an oblique plane than a plane orthogonal to the axis, it would be detected during this step. The other two limitations are eliminated by the Margin Algorithm. It also has an advantage over multivariate decision trees in that the attributes are not processed into features.

The upper bound for the Margin Algorithm is from $O(kn)$ for the global version to $O(kn + k\ln k)$ for the local version, where k is the number of attributes and n is the number of samples.

Case 3: heavy overlap

Depending on the structure of the input data and the algorithm chosen, accuracy will vary for the case of heavy overlap. If the data for all classes is uniformly

interspersed, no algorithm will accurately predict the classes. If one class is enclosed by another, such as a box inside another box, the accuracy can be quite high.

For the case of heavy overlap, the Box Algorithm bears a superficial resemblance to *k*-Nearest Neighbor in that it relies on points in the same class being close to one another. *Boxes* are used to classify the points. For the simplest type of *box*, i.e., a *cube*, the upper bound is $O(kn)$, where k is the number of attributes and n is the number of samples. The results are comparable to those where an *n*-dimensional sphere is used to classify each class.

To gain better accuracy, an *asymmetric box* is formed, but at computational increase. This is on the order of $O(j^{2k}n)$, where j is the number of steps in the k loops required. The increase in accuracy versus cost is justified only for special cases where accuracy is extremely important. For all types of boxes, the computational cost during training for m classes can be reduced by a factor of m if the algorithm is run on parallel processors.

The complexity of Box when using the *cube* shape is less than that of *k*-NN, i.e., is $O(dn^2 + n\ln n)$, where d is the number of attributes and n is the number of samples.

1.5 ORGANIZATION OF THIS THESIS

The Paired Planes Classification Procedure is presented in Chapter 2. Each of the algorithms (Hyperplane Algorithm, Margin Algorithm, and Box Algorithm) used for the three cases of overlap between two classes is explained. Pseudocodes and corresponding system diagrams are provided for the classification procedure and each component algorithm. The theoretical properties of the Paired Planes Classification Procedure are also presented in Chapter 2. Pseudocode and bounds for complexity and training for the Hyperplane Algorithm, Margin Algorithm, and Box Algorithm and the total classification procedure are given.

Chapter 3 takes an in-depth look at the methods used to address the three cases of overlap between two classes. Two heuristics of class order are tested for both binary and multi-class problems. In addition, the effect of rotating of the axes on the accuracy of classification is tested through comparison between the Margin Algorithm and the Hyperplane Algorithm. Artificial datasets are used to explore the behavior of the proposed algorithm. The results of this study have been published in [26] and [27].

In Chapter 4 the methods used to address the three cases of overlap between sets on artificial datasets are tested on real world datasets for both binary and multi-class problems. Our results are compared to results obtained by Support Vector Machines, univariate (C4.5) and multivariate (CART) decision trees, and the *k*-Nearest Neighbor algorithm.

Chapter 5 concludes the dissertation with a critique of the Paired Planes Classification Procedure and with a discussion of future work.

Chapter 2:

The Paired Planes Classification Procedure

2.1 INTRODUCTION

For the two-class problem in the case when the two classes overlap partially, one may consider the sample space divisible into three regions: Class *1*, Class *2*, and a region of overlap in which classification is at best ambiguous. By contrast to the region of overlap, the other two regions can be classified accurately and often by considering only a subset of the attributes. This study takes advantage of this observation in a novel manner. Use of a single decision surface has proven to be a tenable idea that provides good classification with reasonable cost. Our algorithm uses pairs of parallel decision surfaces to classify. There are three separate but related steps (Hyperplane Algorithm, Margin Algorithm, and Box Algorithm) to achieve this. Each of these steps is an approach to split the feature space by pairs of parallel planes.

The first step, the Hyperplane Algorithm [30], [31], works best when the overlap between two classes is either zero or small. The approach has some features in common with Support Vector Machines [11] and uses all attributes simultaneously. The decision surfaces are perpendicular to the vector going from the mean of Class *1* to the mean of Class *2* as computed from the training points. It returns the accuracy of classification of each class and an estimate of the overlap of each class in the region of confusion. If the

overlap is moderate or heavy and therefore the classification accuracy is poor, the amount of overlap is used to decide which of the two other steps will be used next.

In case of moderate overlap, the second approach, the Margin Algorithm [23], [24], [25], is used. This is a decision tree, which results in a disjunctive rule of classification. It uses attributes sequentially, as needed, to classify points either as Class *1*, the overlap, or Class *2*. The decision surfaces are perpendicular to the attribute being used. It returns the accuracy of classification of each class.

In case of heavy overlap or when moderate overlap does not result in an acceptable accuracy, the third approach, the Box Algorithm [26] [27], is used. This is a decision tree that works by a series of conjunctions. It uses all attributes to classify points either as Class *1*, Class *2*, or an unclassifiable region (which is not an overlap region). As with algorithms that create an *n*-dimensional ball about the mean of a class, this algorithm creates an *n*-dimensional box about the mean of a class. It returns the accuracy of classification of each class. The decision surfaces produced are orthogonal to the attributes being used.

The amount of overlap between classes determines the order in which these algorithms are used as steps in a classification procedure. More precisely,

1. No or little overlap (approximately 5% of the data): The Hyperplane Algorithm alone can serve as the classifier. In addition, estimating the overlap is a by-product of classification by this algorithm. Therefore, this algorithm is always used as a first step in the classification procedure.
2. Moderate overlap (approximately 5% to 35% of the data): The Margin Algorithm is invoked as the second step of the classification procedure for this case. Moreover, if

the Hyperplane Algorithm does not return the required/expected accuracy when the overlap is slight, the Margin Algorithm is invoked.

3. Heavy overlap (35% or more of the data): When the classes have heavy overlap, such as when one class is completely inside another, the Box Algorithm is invoked as the second step of the classification procedure. In addition, if the Margin Algorithm fails to deliver the required/expected accuracy, the Box Algorithm is invoked as the third step of the classification procedure. Order of classes in testing classification is important here.

Obviously, if the overlap is too great, none of these algorithms (as with most other approaches) will be able to classify in the original feature space accurately.

2.2 NOTATION AND TERMINOLOGY

For all steps, we initially consider the two-class problem. This is later extended to the multi-class problem, but here the terminology is in terms of the two-class problem.

The following notation and terminology is used throughout this thesis.

- A denotes the acceptable accuracy input to the Hyperplane Algorithm.
- A_H, A_M, and A_B, denote the accuracies returned by the Hyperplane Algorithm, the Margin Algorithm, and the Box Algorithm, respectively.
- *o* denotes overlap as estimated by the Hyperplane Algorithm.
- **N** denotes a vector from the mean of Class *1* to the mean of Class *2*, as computed from the training data.
- **P** denotes a hyperplane, used as decision surface, generated in the training phase of the Hyperplane Algorithm.

- **J** and **K** denote the hyperplanes, used as decision surfaces, determined by the Hyperplane Algorithm.
- θ denotes the angle between a hyperplane and the vector **N**.
- dp denotes the dot product.
- μ_i , $i \in \{1, 2\}$, denotes the mean, where the values for i indicate the class (*1* or *2*); σ_i is the standard deviation for this class.
- μ^k_i , $i \in \{1, 2\}$, denotes the mean value of attribute k corresponding to the training set for Class i ; σ^k_i is the standard deviation for this.
- η^k_i, $i \in \{1, 2\}$, denotes the learning constant for attribute k corresponding to the training set for Class i.
- η_i, $i \in \{1, 2\}$, denotes the learning constant corresponding to the training set for Class i.
- $a_k = \min (\mu^k_1 + \eta^k_1 \sigma^k_1 , \mu^k_2 - \eta^k_2 \sigma^k_2)$; $b_k = \max (\mu^k_1 + \eta^k_1 \sigma^k_1 , \mu^k_2 - \eta^k_2 \sigma^k_2)$
- $m_k = (a_k , b_k)$, where m_k is the local (along attribute k) margin.
- $m_k = (a_k , b_k)$ as above where η^k_1, η^k_2 is replaced by η_1, η_2 and m_k is the global margin for the k^{th} attribute.
- c^Y_k = mean/median for the k^{th} attribute in class Y, $Y \in \{1, 2\}$
- $a_k^Y = c^Y_k - \eta^Y_k \sigma^Y_k$; $b_k^Y = c^Y_k + \upsilon^Y_k \sigma^Y_k$
- w_k^Y , $w_k^Y = [a_k^Y, b_k^Y]$, denotes the side of the k^{th} attribute in class Y, $Y \in \{1, 2\}$.
- b^Y , $b^Y = w_1^Y$ x … x w_n^Y , where x denotes the Cartesian product, denotes the box for class Y, $Y \in \{1, 2\}$.
- \ denotes set difference.

- $X = \{x_1, \ldots, x_n\}$ denotes the collection of attributes of interest for a two class classification problem.
- TRAIN denotes the training set for this problem, i.e., TRAIN = $\{(x_1, \ldots, x_n, y)\}$, $y \in \{1, 2\}$, where the values for y indicate the class (*1* or *2*) that the vector $(x_1, \ldots, x_n)$ belongs to.
- X_k denotes the domain of the attribute x_k as represented in TRAIN.
- X^y_k denote the domain of the attribute x_k represented in class y.
- Finally, $X_k = X^1_k \cup X^2_k$.

2.3 PAIRED PLANES CLASSIFICATION PROCEDURE, A THREE-STEP PROCEDURE

We consider three cases as shown in Figures 2-1, 2-2, and 2-3.

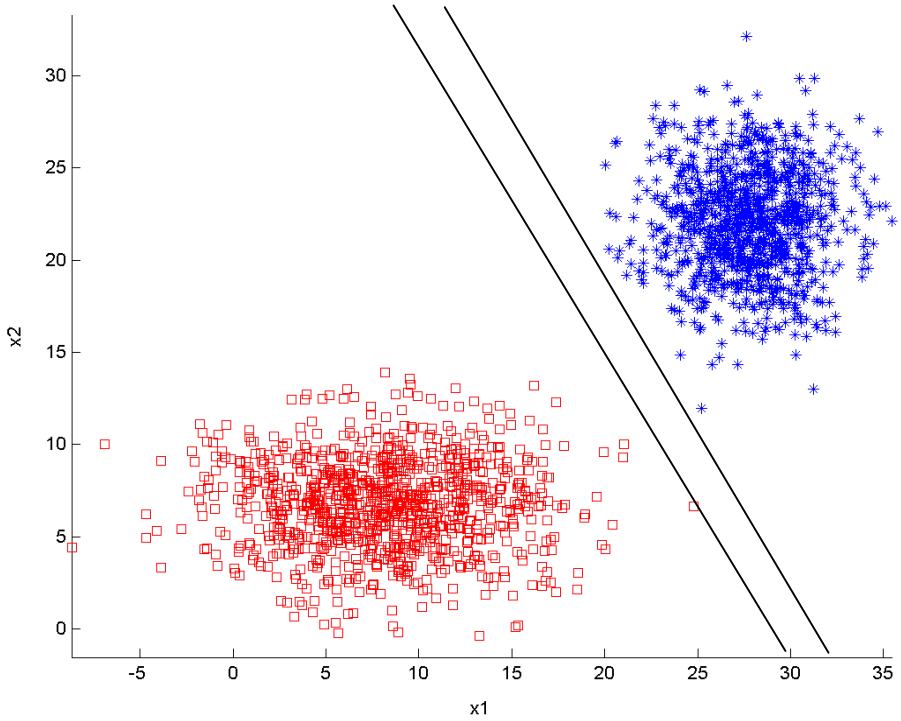


Figure 2-1. Hyperplanes to Separate Directly.

1. No overlap or very small (0 – 5%) overlap, Figure 2-1. The region between parallel hyperplanes (lines in two dimensions) can be an area of overlap or, if classes are separable, without any points. Points in this region cannot be classified by such hyperplanes. The objective of the algorithm is to find the two hyperplanes shown in this figure.

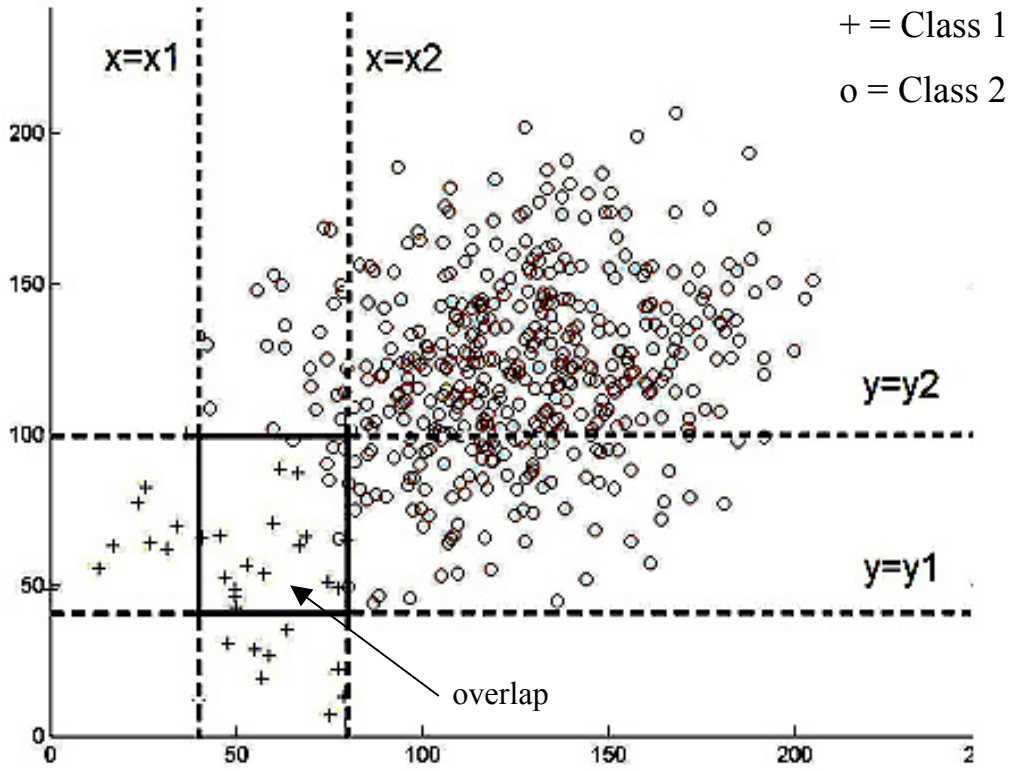


Figure 2-2. Hyperplanes to Form Margins.

2. Moderate (5% – 35%) overlap, Figure 2-2. The region in the solid box is the area of overlap. Points in this region cannot be classified by this algorithm. The dashed lines define *margins (of overlap)* for each attribute. The *margin* is simply an interval along an attribute where the classes overlap. Different orders of the attributes (x1, x2) can be used with different classification accuracy. Therefore,

attribute order becomes important. The objective of the algorithm is to determine the *margin* obtained from the two hyperplanes (dashed lines) corresponding to each attribute, as well as the *best* attribute order for classification.

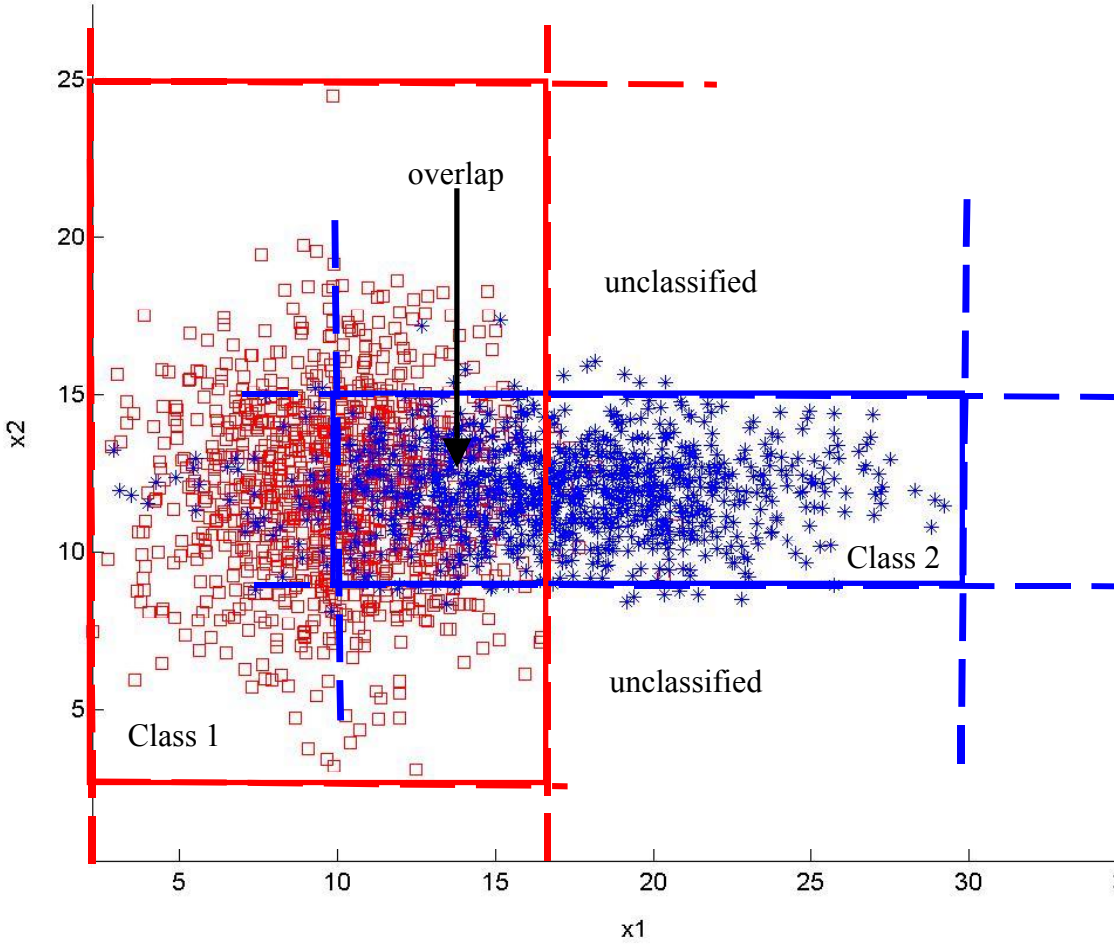


Figure 2-3. Hyperplanes to Form Boxes.

3. Heavy (35% or more) overlap, Figure 2-3. Some points in this region of overlap (intersection of two boxes) can be classified correctly while others will produce errors. The amount of error will also depend on the class order of classification. Therefore, class order becomes important. Class order is determined by the overlap estimate from a prior step. In addition, there will be an *unclassified*

region whose points cannot be classified. The objective of the algorithm is to determine the sides of the *box* obtained from the two hyperplanes (dashed lines) corresponding to each attribute, as well as the *best* class order for classification.

Although, for presentation purposes, these methods are introduced as separate algorithms, it should be noted that they have an underlying common approach: Each uses parallel hyperplanes to divide the sample space into three regions – Class *1*, Class *2*, and a region where the points are unclassifiable. These methods differ in the way these parallel hyperplanes are obtained.

The classification procedure can be summarized as follows: Given desired accuracy A:

1. Apply the Hyperplane Algorithm. Returns overlap estimate o and accuracy A_H.
   - If $A_H \geq A$, return accuracy A_H and hyperplane parameters.
   - If $A_H < A$ and o = moderate, apply the Margin Algorithm to the whole set to obtain accuracy A_M.
   - If $A_H < A_M$ and o = heavy, apply the Box Algorithm to obtain accuracy A_B.
2. The Margin Algorithm ($A_H < A$): Returns accuracy A_M.
   - If $A_M > A_H$, return accuracy A_M and margin parameters.
   - If $A_M \leq A_H$, apply the Box Algorithm to obtain accuracy A_B.
3. The Box Algorithm ($A_H < A$): Returns accuracy A_B.
   - If $A_B > A_H$, return accuracy A_B and box parameters.
   - If $A_B \leq A_H$, return accuracy A_H and hyperplane parameters

The classification procedure described above is shown in the diagram of Figure 2-18 of section 2.7.

2.4 THE HYPERPLANE ALGORITHM

As can be noted (and illustrated in Figure 2-2), the Margin Algorithm, as most univariate decision trees, generates hyperplanes orthogonal to the axes. However, when classes are not separable by such hyperplanes but still separable, the Margin Algorithm will fail to detect this. So, the Hyperplane Algorithm is meant to detect such cases and to estimate the overlap (if any).

Moreover, it turns out that if the amount of overlap is sufficiently small, the Hyperplane Algorithm by itself is the classification tool. If not, by providing a measure of class overlap o, it allows us to choose which of the two remaining algorithms – the Margin or the Box – to use to classify most accurately. In other words, it can be used as a heuristic to decide whether to use the Margin or the Box Algorithm.

Let **N** denote the vector connecting the means, μ_i (i = 1, 2), of the two classes, as obtained from the training data. The algorithm looks for a hyperplane perpendicular to **N**. It is possible to construct an infinite number of such hyperplanes, any one of which can be used to split the sample space into two regions.

By use of the dot product, we can determine where each point of the training data is with respect to the hyperplane perpendicular to the head of **N**. The vector **N** and the hyperplanes **J** and **K** used to separate the classes are shown in Figure 2-4.

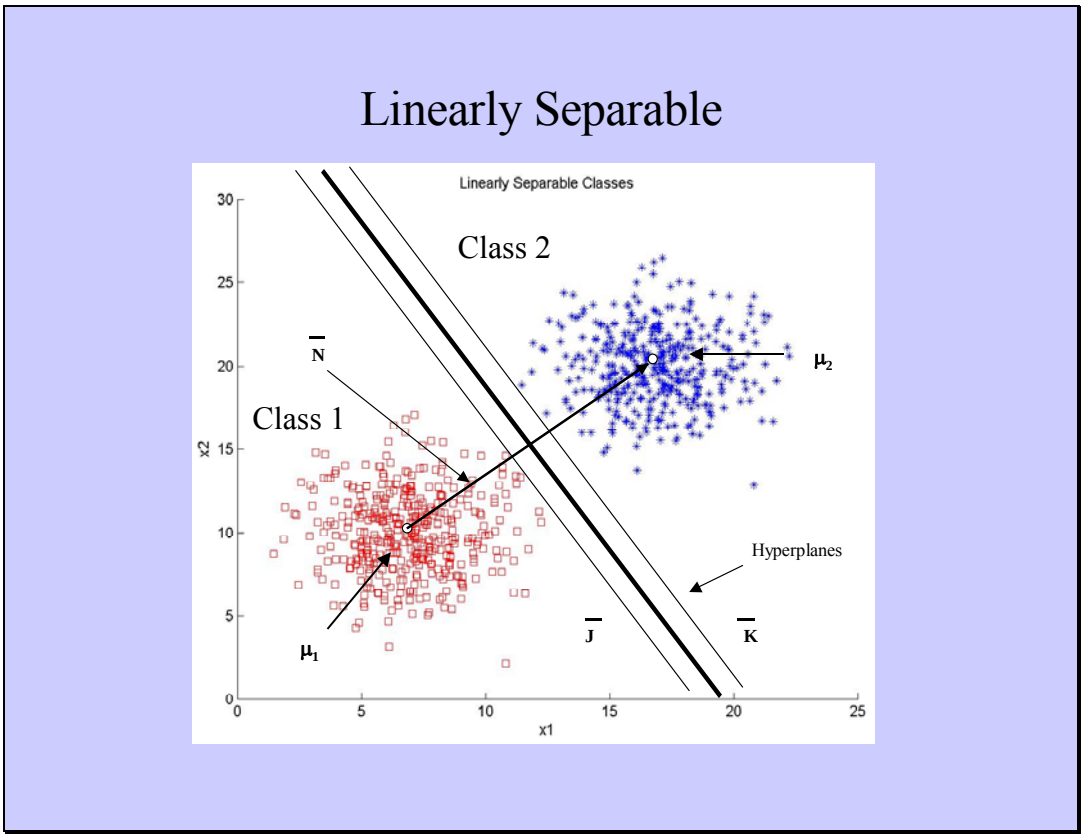


Figure 2-4. How two parallel hyperplanes might be used to separate two classes.

The final hyperplanes **J**, **K**, and decision hyperplane are obtained as follows:

We form a vector **V** from each point in the data set to the head of **N**. For $\theta = \frac{\pi}{2}$, $\cos\theta =$ 0 and $\frac{< N, V >}{\| N \| \| V \|} = 0$. Therefore, positive values of <**N**,**V**> will denote points on one side of the hyperplane, negative values will denote points on the other side of the hyperplane. Construction of the vector **N** is shown in Figure 2-5.

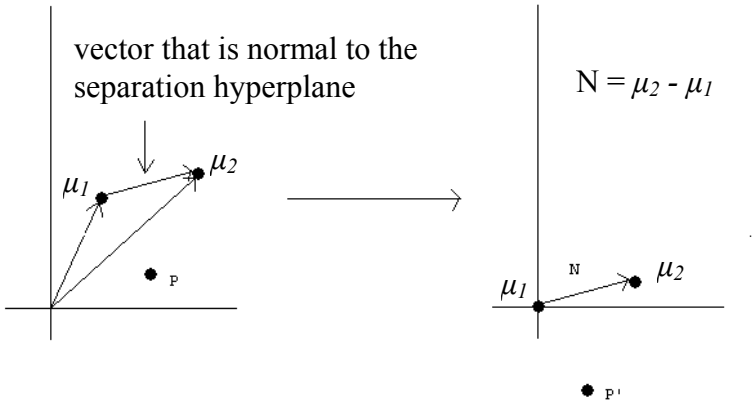


Figure 2-5. Creation of vector **N** and translation of the training dataset.

Next, we consider two subcases - separable classes and non-separable classes.

Separable classes:

For each such hyperplane, we can ascertain whether it forms a decision surface that completely separates all points in one class from the other class. We start with the hyperplane **P** going through the head of **N** at μ_2. We check to see if all the points in Class *2* are on one side of the plane and all points in Class *1* are on the other side. If not, we successively shorten the vector **N**, checking at each iteration to see if **P** successfully separates the two classes without error. Let **K** denote the first hyperplane that separates Class *2* from Class *1* without error. We continue shortening the vector **N**. Let **J** denote the last hyperplane that separates Class *2* from Class *1* without error.

For such **K** and **J**, the final decision surface is the hyperplane midway between **K** and **J** and parallel to them. This ensures maximum generalization with respect to the hyperplane given this training data.

Non-separable classes:

On the other hand, if we cannot find **K** and **J** such that perfect accuracy is assured, this process is still useful. It is used to maximize the region of classification for each class and to estimate the degree of overlap as follows.

First, we seek to maximize the region where we can correctly classify points for Class *2* while also correctly classifying *all* points of Class *1*. During the process of finding **K**, we check at each iteration to see if **P** has correctly classified all points for each class. In the event of a point from Class *2* being incorrectly classified and on the same side of **P** as μ_2, we back up one step and use the previous hyperplane as **K**. This is the *last* hyperplane such that there were no errors of classification from it towards μ_2. Note that the first hyperplane **P**, which goes through μ_2, may classify one or more points from Class *2* incorrectly. In that case, **K** is defined to be the starting hyperplane going through μ_2.

Second, we seek to maximize the region where we can correctly classify points for Class *1* while also correctly classifying *all* points of Class *2*. We continue *marching* the hyperplane **P** towards μ_1. During the process of finding **J**, we check at each iteration to see if all points for each class have been correctly classified. When a point from Class *2* is incorrectly classified, we continue. We choose the *first* hyperplane that makes no errors of classifications for points in Class *1* and on the same side of **P** as μ_1. This is the *first* hyperplane such that there were no errors of classification from it towards μ_1. Note that such a hyperplane may not exist, in which case, **J** is defined to be the hyperplane going through μ_1.

In both subcases, we found the maximum region for each class where classification is without error for that class. If the classes are separable by this algorithm,

J and **K** are used to generate an optimal decision surface. On the other hand, if the classes are not separable by this algorithm, **J** and **K** are used to estimate the overlap (as computed from the training data). By counting the number of points between **J** and **K**, we can estimate the extent of overlap (as a percentage of the training data) for each class. This in turn allows us to choose step 2 or step 3, depending upon the degree of overlap.

The approach can be adjusted to take into account values of θ different from $\frac{\pi}{2}$. Stepping through the values for θ, the algorithm will find **J**, **K**, and θ for best accuracy. Figure 2-6 shows a hypothetical example of this.

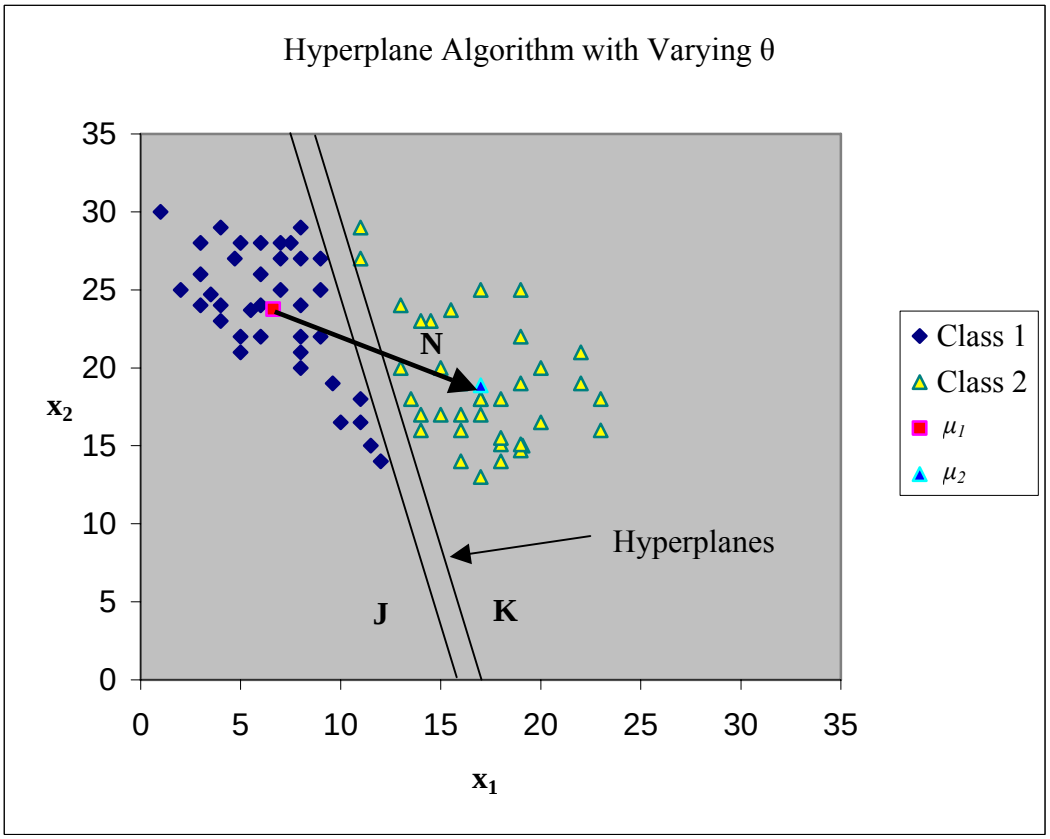


Figure 2-6. Parallel hyperplanes *not* perpendicular to the vector **N** separate the classes.

By relaxing the restriction that the pair of hyperplanes is parallel to one another, as well as taking into account values of θ different from $\frac{\pi}{2}$, separating surfaces shown in Figure 2-7 can be generated. In such a case, the hyperplanes would intersect and the separating surfaces are more complex. Such an investigation was beyond the scope of this study.

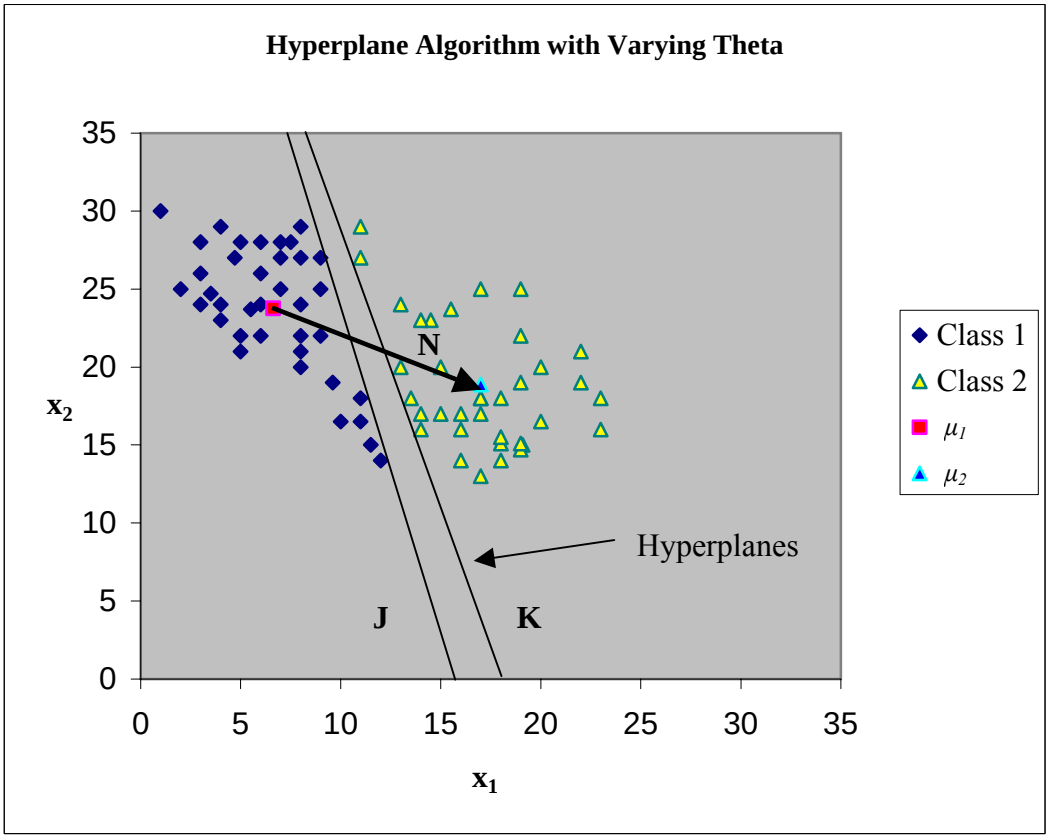


Figure 2-7. Restriction of Parallel Hyperplanes Lifted.

Alternatively, a certain number of errors for either or both classes could be deemed acceptable. For instance, in the case of separating benign cancers from

malignant cancers, it may be acceptable to a user that 5% of benign cancers are classified incorrectly and 0% of malignant cancers are classified incorrectly.

Such criteria can be easily incorporated in the algorithm. In addition to the relevance to the user, allowing a certain amount of error in the training stage may result in improved accuracy in the test data set. It should be noted that current algorithms using SVMs cannot add such criteria. A soft margin SVM [32], by use of *slack variables*, accepts some error in classification. However, the number of errors and their distribution (Class *1* or Class *2*) is controlled only indirectly through the total amount of *slack* (amount of error).

2.5 THE MARGIN ALGORITHM

The basic idea for the Margin Algorithm is as follows: For a particular attribute, one can derive hyperplanes that quickly and accurately classify some/many points of the training set. A formal definition of *margin* that allows some errors of classification follows.

2.5.1 MARGINS

We discuss the formation and use of the margin by the two-class problem. Two versions of the algorithm are considered - the *global* and the *local*. Constructing the margin takes into account a tradeoff between the number of correct classifications for a particular class and the total number of correct classifications. A predetermined number of steps are used. The margin with the highest accuracy for the training set is chosen.

To define the margins the following quantities are introduced: μ^k_1, μ^k_2 mean values of attribute k corresponding to the training set for Class *1* and Class *2*,

respectively, and $\mu^k_1 < \mu^k_2$ for all k; σ^k_1, σ^k_2 standard deviations for these classes along attribute k; $0 < \eta^k_1, \eta^k_2 < N_k$ and $0 < \eta_1, \eta_2 < N$.

Definition 2-1:

(a) The local (along attribute k) margin m_k is the interval (a_k , b_k) where

$$a_k = \min (\mu^k_1 + \eta^k_1\sigma^k_1 , \mu^k_2 - \eta^k_2\sigma^k_2) \qquad (2\text{-}1)$$

$$b_k = \max (\mu^k_1 + \eta^k_1\sigma^k_1 , \mu^k_2 - \eta^k_2\sigma^k_2)$$

(b) The global (along all attributes) margin m_k is defined by the interval (a_k , b_k) as in (2-1) where η^k_1, η^k_2 is replaced by η_1, η_2, respectively.

The margins are found for each attribute, using η^k_1 and η^k_2 for the local version and using η_1 and η_2 for the global version. Their values determine both the accuracy and the speed of the convergence of the algorithm. It can be seen that X_k can now be written as $X_k = (X^1_k \setminus X^2_k) \cup m_k \cup (X^2_k \setminus X^1_k)$, where \ denotes set difference.

The number of standard deviations for each class determines the margin. The underlying class distribution is considered Normal (the Central Limit Theorem [4] justifies this assumption for large sets). This means that with high probability the class values fall within three standard deviations from the class mean, allowing us to conclude that for all practical purposes, all points in the training set have been considered.

As the margins are marched, they approach each other and eventually pass one another. The margins are updated according to Definition (2-1) on successive values of η. Basically, the mechanism of constructing the margin *marches* the quantities a_k , b_k away from the class means for attribute k by considering successive values of η. The differences between the local and global versions lie in how and when the values η^k_1, η^k_2 or η_1, η_2 are updated for each attribute.

The constants η^k_1, η^k_2 and η_1, η_2 can be viewed as the learning constants of the corresponding algorithms. Updating for each attribute before using the next attribute results in the *local* version. This update proceeds whenever the accuracy given by attribute k with new values of η^k_1, η^k_2 is better than that obtained with previous values of these constants. Updating after all attributes have been used in an increment results in the *global* version. This is an update whenever the total accuracy given by all attributes in combination using new values of η_1, η_2 is better than it was with previous values of these constants.

The learning constants η^k_1, η^k_2 / η_1, η_2 can be varied independently with different increments. The pseudocode for the Margin Algorithm is found in Chapter 3. The placement of the updating step for the values of η is the only difference between the pseudocode for the two versions. In the global version, it appears after all attributes have been used in an iteration. In the local version, it appears after each attribute has been used in an iteration.

On the local version of the Margin Algorithm, the order of attributes is important and it is determined at training by cross validation.

Assuming that m_k (the margin for the k^{th} attribute) has been found, the classification of data points is done as follows.

Two Classes

When scanning the training data set in a given direction, all of the examples of Class *1* appear to one side of the margin, followed by the margin, followed by all of the examples for Class *2*. (Without loss of generality, assume that the mean of Class *1* is to the left of

the mean of Class *2* for each attribute.) If $m_k = (a_k, b_k)$, then the class assignment for the data point x is given by the rule:

$$Class(x) = \begin{cases} 1 & \text{if } x \le a_k \\ 2 & \text{if } x \ge b_k \\ \text{none} & \text{if } x \in m_k \end{cases} \qquad (1)$$

At first glance, the Margin Algorithm appears to produce a Top-Down Inductive Decision Tree (TDIDT) such as those produced by C4.5. However, this is not really the case. More precisely, the Margin Algorithm

- uses a *structural geometric criterion* to construct a tree while ID3/C4.5 algorithms use an information theoretic criterion to construct a tree.
- is very transparent, using each attribute at most once in the tree to classify a point. TDIDT has a series of rules to make the classification decision.

It is worth noting the structure of the decision tree produced by the Margin Algorithm: a tree of depth equal to the number of attributes in which each node has exactly (m+1) children (where m is the number of classes). To illustrate the characteristics of the tree produced by the Margin Algorithm and C4.5-type of algorithm, consider the trees produced for the well-known Iris data set [21].

Compare the tree made by each method. The C4.5-type tree [33] shown in Figure 2-8 uses the attribute PL (petal length) more than once. The Margin Algorithm does not. It uses PL once to decide if the point is classifiable by this attribute as one of the three classes or unclassifiable. It uses a series of disjunctions to classify.

The C4.5-tree checks initially whether PL < 26.0. If so, it is classified as Iris-Setsosa, otherwise, PW (petal width) is checked. Depending on the value of PW, it is again checked for PL of differing values, and then against SW (sepal length) or SL (sepal length). This is a series of conjunctive statements used as rules. This tree uses two subtrees to separate class Versicolor from class Virginica. Each subtree represents a different set of conjunctions. Measurements are in millimeters.

According to this tree the Iris-Virginica class is described by the following rules:

i) PL ≥ 26.0 *and* PW < 17.5 *and* PL < 49.5 *and* SL < 49.5

ii) PL ≥ 26.0 *and* PW < 17.5 *and* PL ≥ 49.5 *and* SW < 26.5

iii) PL ≥ 26.0 *and* PW ≥ 17.5 *and* PL < 48.5 *and* SL < 59.5

iv) PL ≥ 26.0 *and* PW ≥ 17.5 *and* PL ≥ 48.5

Similar rules are produced for Iris-Versicolor and Iris-Setsosa. Note that the rules i) – iv) use multiple tests of the same attribute(s).

By contrast, the tree produced by the Margin Algorithm (shown in Figure 2-9) is of depth four (because there are four attributes) and with four children for each node (because there are three classes) tests PL only once. Moreover, as can be seen from Figure 2-9 a classification decision can be made at any level of the tree based on the attributes considered up to that level.

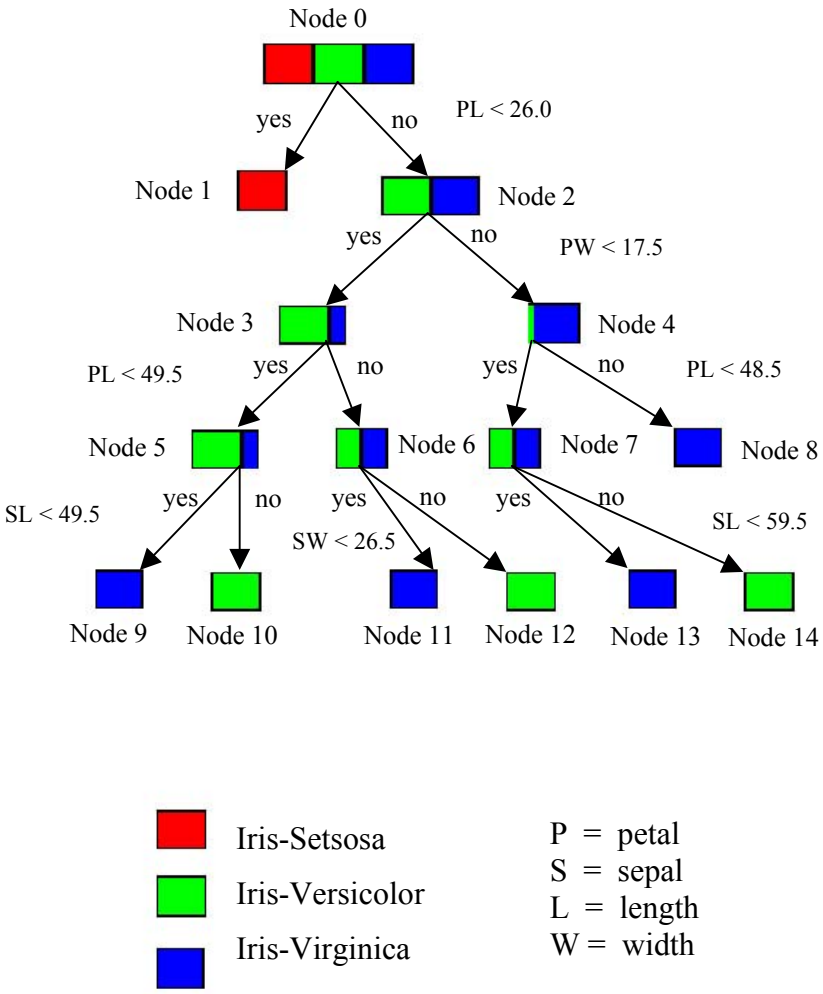


Figure 2-8. C4.5-type's Decision Tree for the Iris Dataset [33].

The tree produced by the Margin Algorithm is much simpler and produces much simpler rules. Measurements are in centimeters.

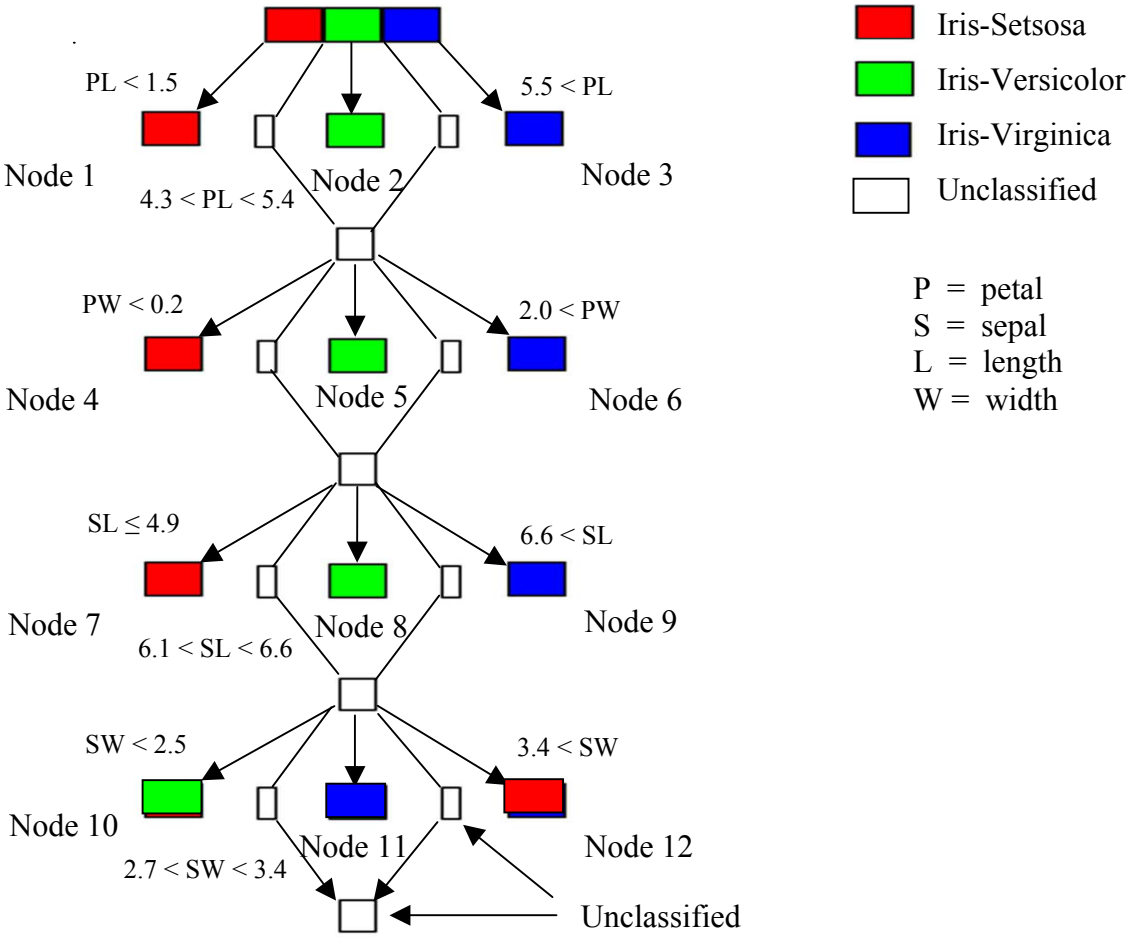


Figure 2-9. Margin's Decision Tree for the Iris Dataset.

The Iris-Virginica class is described by the following simple rule:

[PL > 5.5 *or* PW > 2.0 *or* SL > 6.6 *or* 2.7 < SW < 3.4].

Visualization of the Process

Figures 2-10 through 2-13 illustrate the algorithm described above. Informally, the x_k-*margin* is the region of overlap between the two classes along the k^{th} attribute. Figure 2-10 illustrates this: There are three regions corresponding to attribute x_1 - Class *1* on the left, Class *2* on the right, and the region of overlap between the two classes.

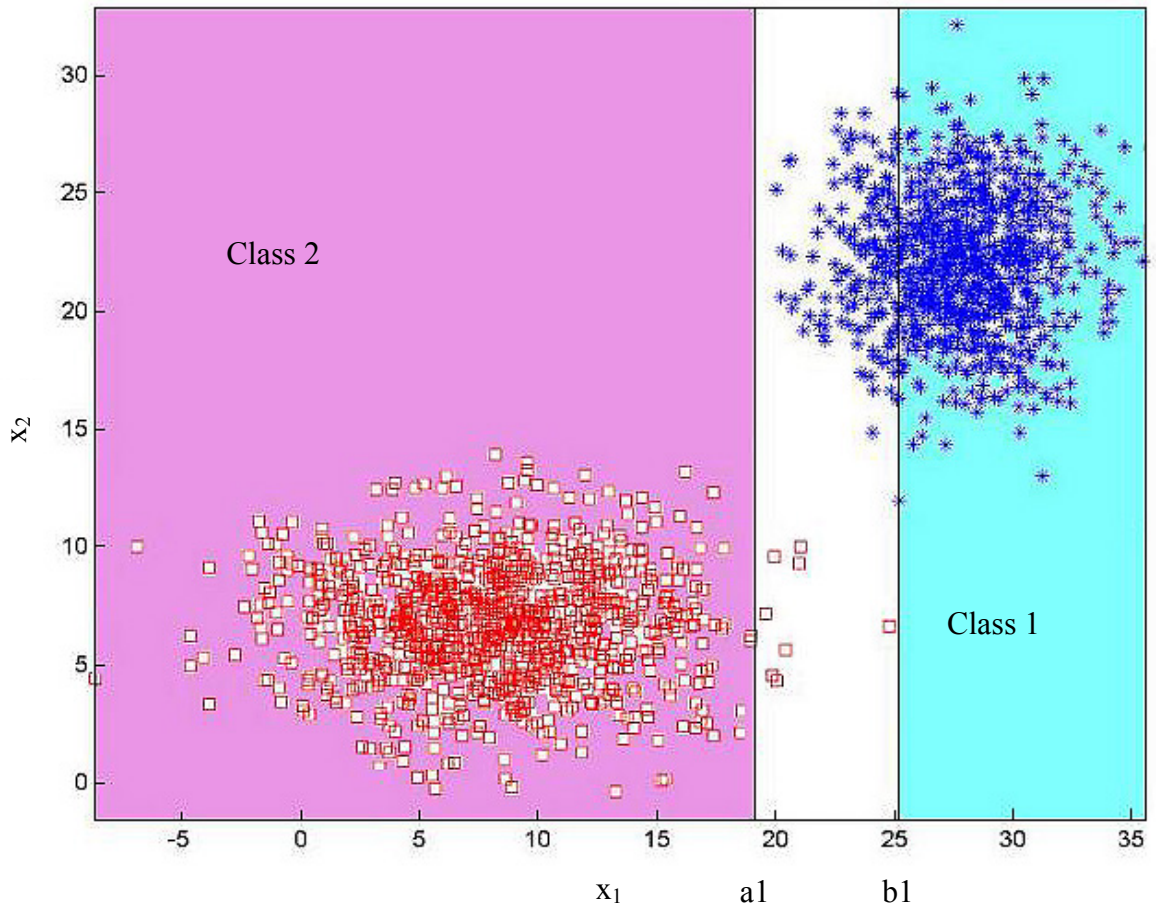


Figure 2-10. The data set is shown with the x_1 margin.

In Figure 2-11, the points classified by x1 have been removed. The data points remaining, i.e., the overlap region is much smaller than the original region. Its points can subsequently be classified by x_2, as shown in Figure 2-12.

The region of uncertainty (the remaining unclassified points) to be classified after using x_2 is smaller yet. Again, the regions to either side of the *margin* are accurately and quickly classified. Only points in the *x_2-margin* that were *also* not classified by the *x_1-margin* are left to classify. Figure 2-12 shows again the overlap region and the correct classification of its points as Class *1* or as Class *2*.

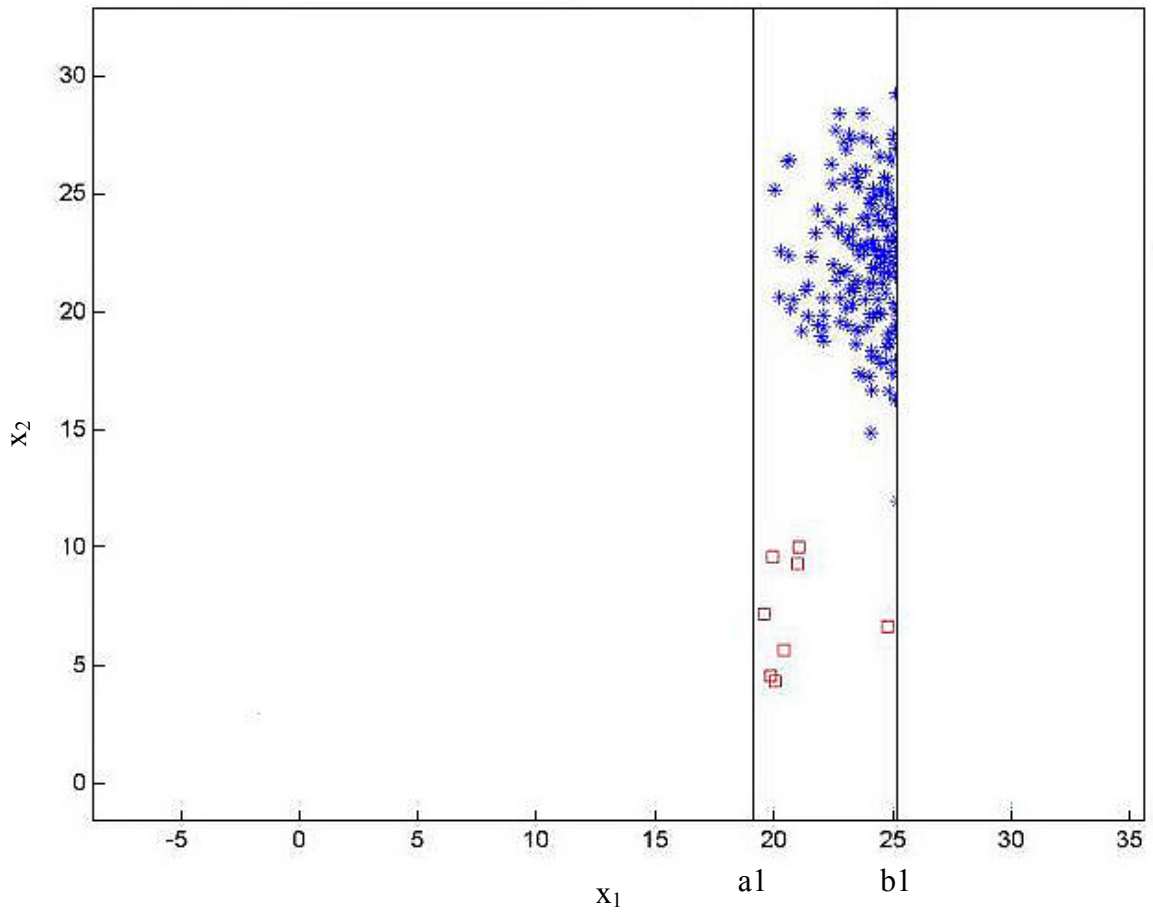


Figure 2-11. The data set for after points classified have been removed.

Eventually, either all points are classified or all attributes are used and a non-empty set of unclassifiable points remains. In the example (illustrated in Figure 2-12) there are no points remaining to be classified. Obviously if the overlap is complex, one will not be able to classify many points. However, many other algorithms acting in this feature space would fail at this point too.

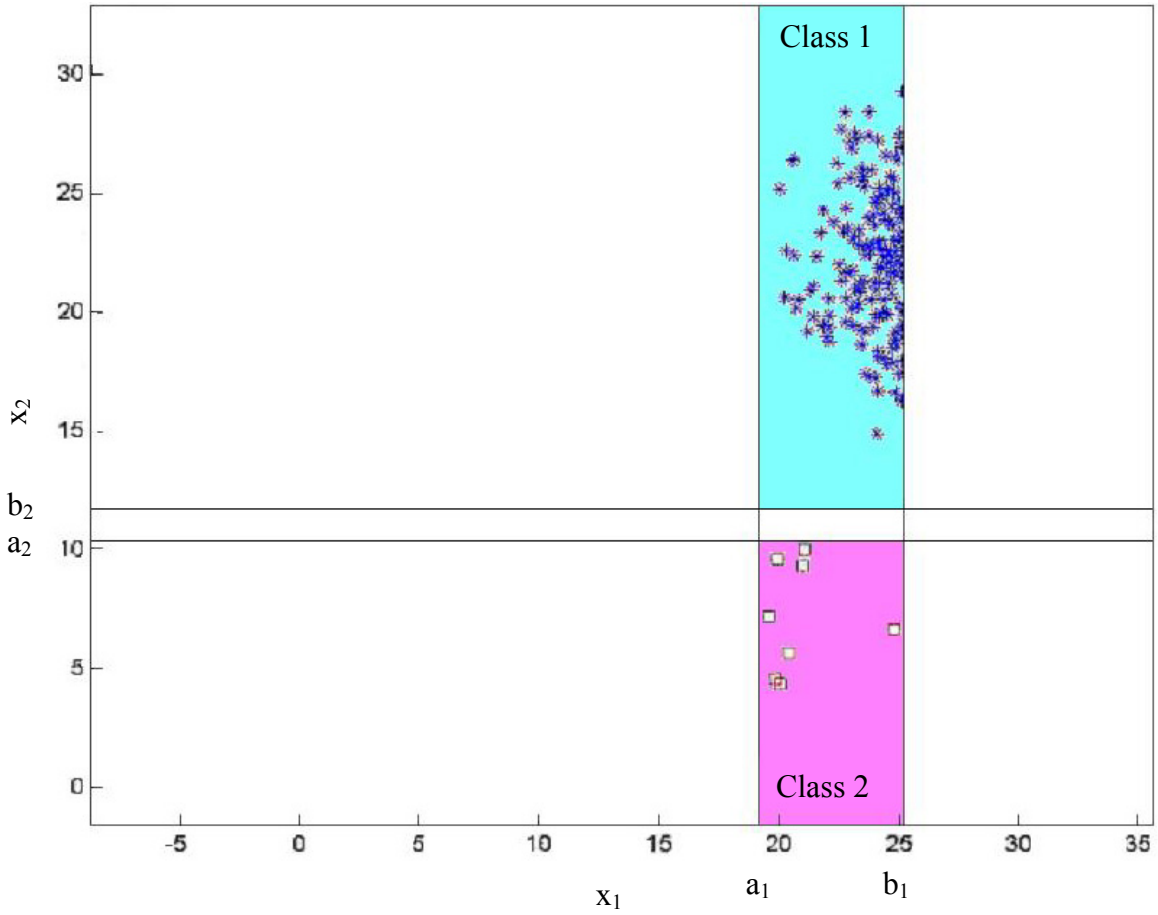


Figure 2-12. The data set is shown after using both the x_1 margin and the x_2 margin. The top region is Class *1* and the bottom region is Class *2*.

In Figure 2-13, the result of using the margins for both attributes is shown. An L-shaped region is created for each of the two classes. The rectangular region that remains in the center is where the unclassifiable points, if any, are left.

Classification rules inferred for this example are:

Class *1*: $(x_1 > b_1) \cup (a_1 < x_1 < b_1 \cap x_2 > b_2)$ Overlap x_1: $a_1 < x_1 < b_1$

Class *2*: $(x_1 < a_1) \cup (a_1 < x_1 < b_1 \cap x_2 < a_2)$ Overlap x_2: $a_2 < x_2 < b_2$

Unclassifiable: $(a_1 < x_1 < b_1 \cap a_2 < x_2 < b_2)$ Overlap: (Overlap x_1) x (Overlap x_2)

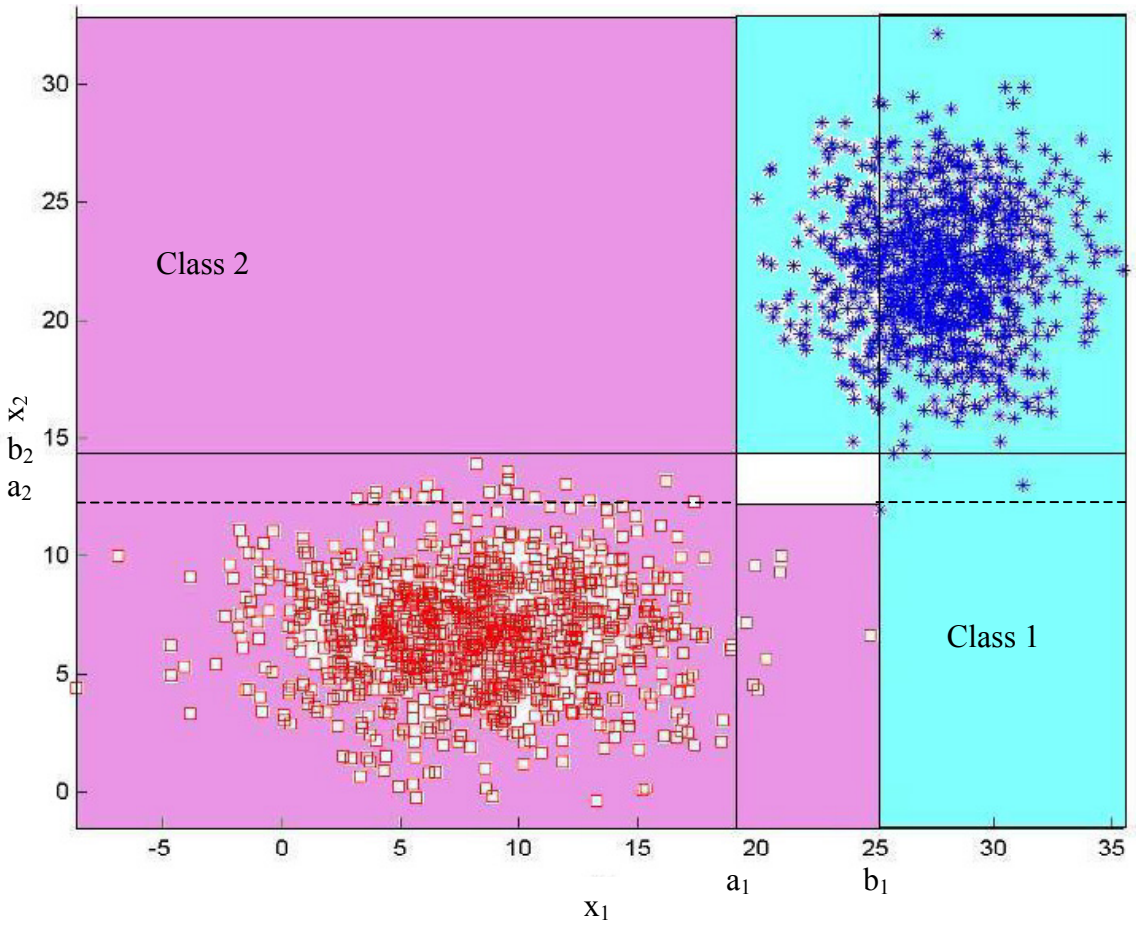


Figure 2-13. Classification Completed.

2.6 THE BOX ALGORITHM

Both the Hyperplane algorithm and the Margin Algorithm (a univariate tree) relied on using parallel lines to define the region of overlap between classes. The regions not in the overlap could be classified easily. However, when one class is entirely within another class, the overlap is complete and the previously developed methods are not sufficient.

Basic Idea

We seek to define the region for a class by pairs of parallel hyperplanes in each dimension resulting in high-dimensional boxes.

For instance, the boxes may start from the means or medians of the classes, as computed from the training set. If the parallel hyperplanes are so that for a given dimension they are equidistant from the start point, a *symmetric box* is produced and the start point is the geometric center of the box. If for all dimensions, *all* pairs of parallel hyperplanes are required to be the same distance from the start point, a *hypercube* is produced. On the other hand, if the equidistance restriction is lifted, an *asymmetric box* (with respect to that point) is produced.

For each attribute, the measure of distance is in units of standard deviation from the start point. The number of standard deviations considered must ensure that almost all attribute values are covered. Usually, this number can be expected to be less than or equal to three.

Width of the Box and Classification

With the notation and terminology of section 2.3 and by analogy to the margin, the quantities that control the behavior of the Box Algorithm are the widths of the box (along each attribute).

Definition 2-2: The width w_k^Y along the k^{th} attribute for class Y is defined as follows.

$$w_k^Y = [a_k^Y, b_k^Y] \quad \text{where}$$

$$a_k^Y = c^Y_k - \eta^Y_k \sigma^Y_k \quad (c^Y_k = \text{mean/median for the } k^{th} \text{ attribute in class } Y) \qquad (2\text{-}2)$$

$$b_k^Y = c^Y_k + \upsilon^Y_k \sigma^Y_k \quad (\sigma^Y_k = \text{standard deviation for the } k^{th} \text{ attribute of class } Y)$$

and where η^Y_k, υ^Y_k vary from 0 to 3.

For $\eta^Y_k = \upsilon^Y_k$, the resulting box, centered at the start point, is *symmetric* along the k^{th} attribute. If $\eta^Y_k = \upsilon^Y_k$ for all k, the resulting box, centered at the start point, is a *hypercube*. Otherwise, if $\eta^Y_k \neq \upsilon^Y_k$ for some k, the box is not centered at the start point.

In scanning the training data set, each class is identified independently of the others. When classifying the testing data, the order of classes is critical for test sets that have one class inside the other, regardless of what kind of box is used. Best order can be inferred by the overlap estimated in the first step (Hyperplane Algorithm).

Assuming that the widths w_k^Y for all attributes have been found, a generic data point x is classified by rule 2 as follows:

$$Class(x) = \begin{cases} 1 & \text{if } x \in b^1 \\ 2 & \text{if } x \in b^2 \\ \text{none} & \text{if } x \notin b^1 \text{ and } x \notin b^2 \end{cases} \qquad (2)$$

where $b^Y = w_1^Y \text{ x } \ldots \text{ x } w_n^Y$ denotes the box with sides $w_1^Y \ldots w_n^Y$ and x denotes the Cartesian product.

As in the case of many classifier-learning algorithms, including the Hyperplane and Margin Algorithms, the Box Algorithm can be modified/adjusted to allow for more errors in the training phase. The objective of doing this is to improve the robustness and overall accuracy of the resulting classifier.

To this end, a *penalty* for each incorrect classification, varying from [0, 1.5] is introduced in the training. A sequence of boxes is generated. For two hypotheses

(boxes), the algorithm choose the one with better overall classification after taking into account the penalty.

In order to achieve better generalization, a *maximal area* heuristic is used: according to this, for equal accuracy, the box of larger area is preferred.

Example 1 illustrates the issues of constructing the boxes for this algorithm.

***Example* 1:** In this example we consider three box cases: a) *too large,* b) *too small*, and c) *reasonable boxes*. An artificial data set is used to illustrate the properties of these boxes.

a) Boxes are too large:

In Figure 2-14, the box is defined by the maximum and minimum values for each class in each attribute. That is, the box b^Y for class Y is given by definition 2-3.

Definition 2-3: The box b^Y for class Y is defined as follows.

$$b^Y = \overset{k}{\underset{i=1}{X}} [\min_{\in xY} x_i^Y, \ \max_{\in xY} x_i^Y] \tag{2-3}$$

While the area for each class is accurately captured, there is a large area of overlap, which leads to high classification error (class dependent).

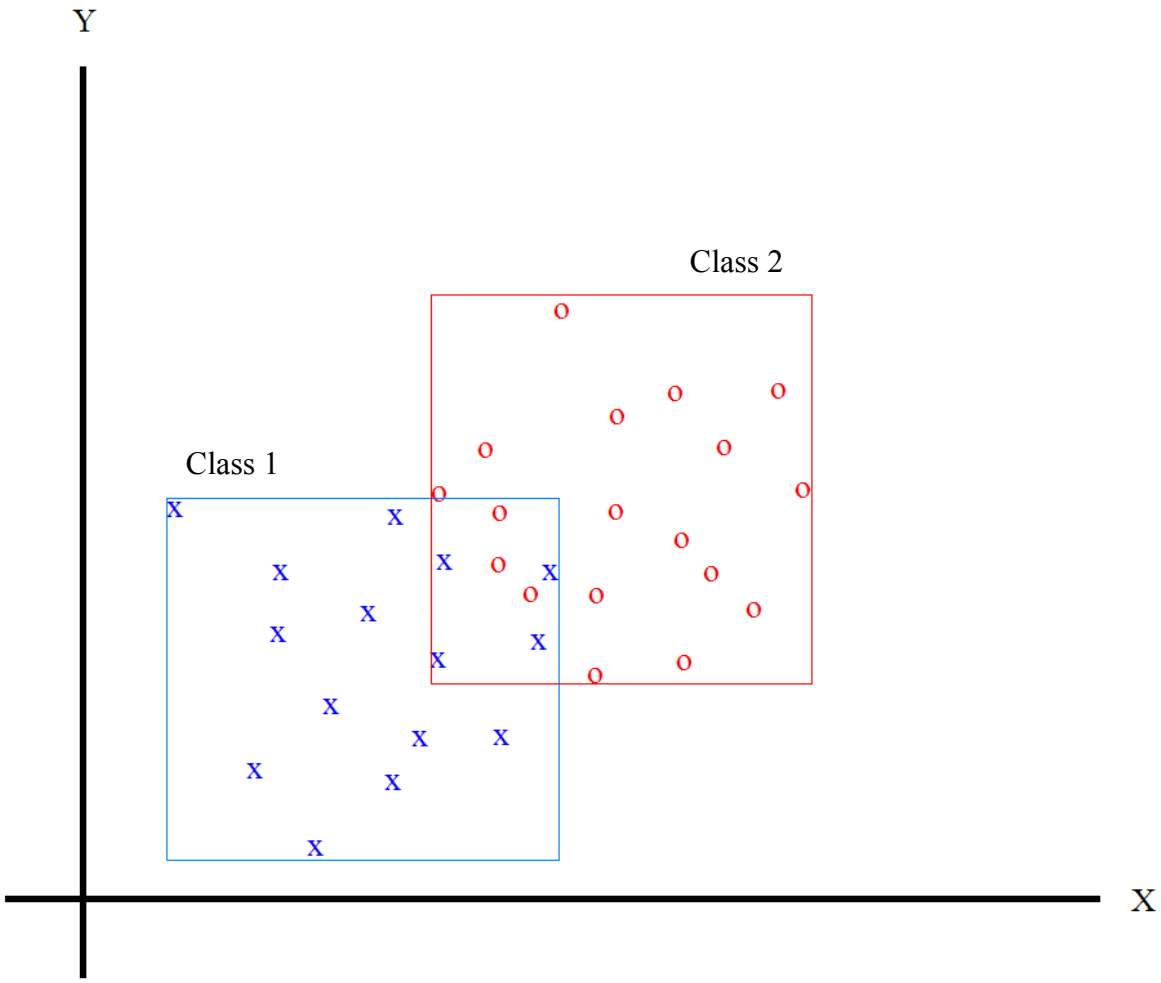


Figure 2-14. Too Large Boxes.

b) Boxes are too small:

Figure 2-15 shows boxes such that all points in them are classified accurately. However, points $x \in b^1 \bigcap b^2$ (not in either of the two boxes) are not classified at all. Therefore, all points classified in a box are correctly classified, but many points remain unclassified.

The boxes in Figure 2-15 cannot be guaranteed to be symmetric with respect to the class mean/median. The point of this example is *not* the construction of these boxes but the fact that such boxes exist and they are too small.

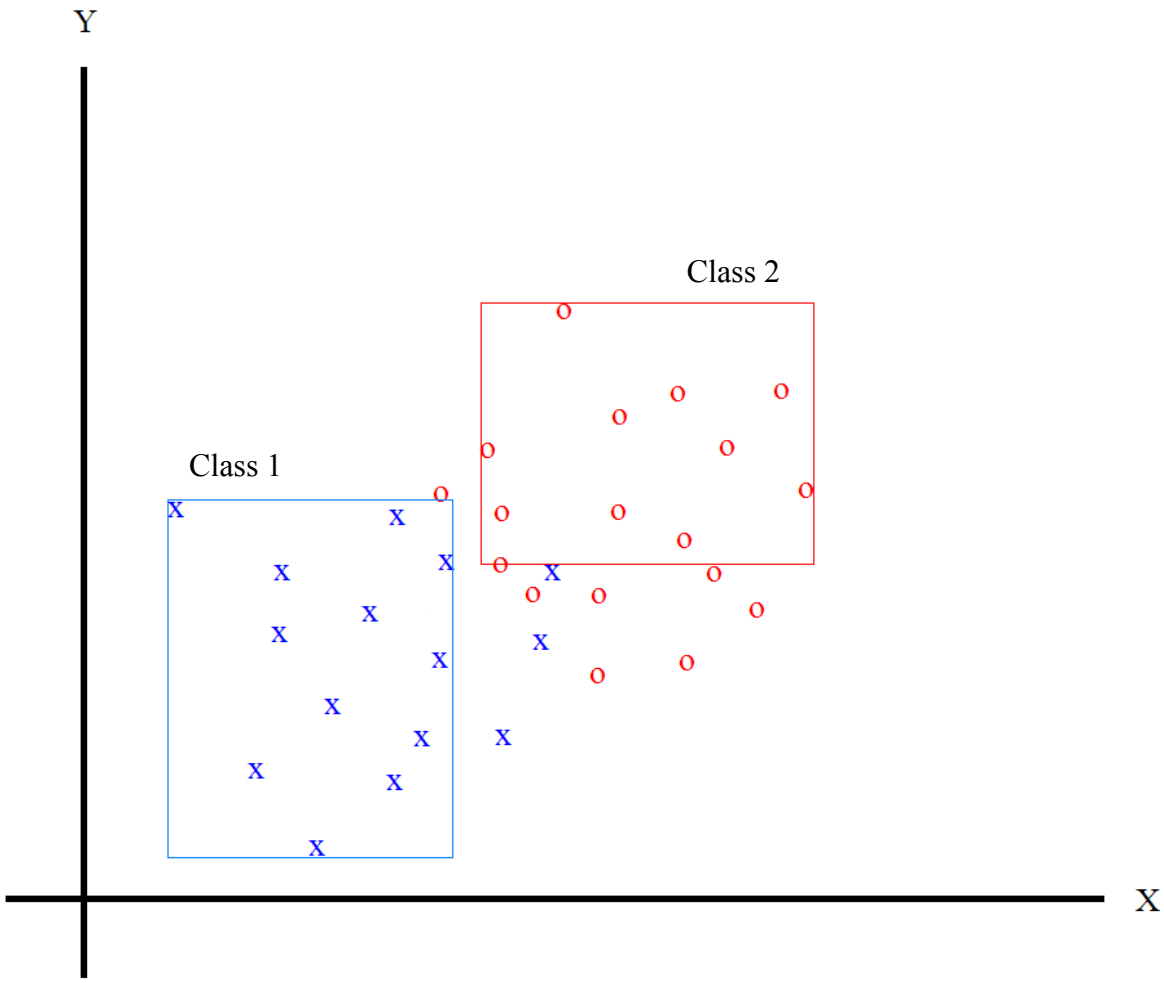


Figure 2-15. Too Small Boxes.

c) Boxes are reasonable:

Figure 2-16 illustrates the case when the boxes are neither too large (as in Figure 2-14) nor too small (as in Figure 2-15). All of the points in one box are correctly classified (the *x*'s), some of the points in the other box (the *o*'s) are incorrectly classified, and two points are not classified. Note that while we show two boxes without overlap, at the final position, the boxes may overlap, especially when we require the box to be symmetric with respect to the start points. In this case, the order in which the boxes are used affects the classification accuracy.

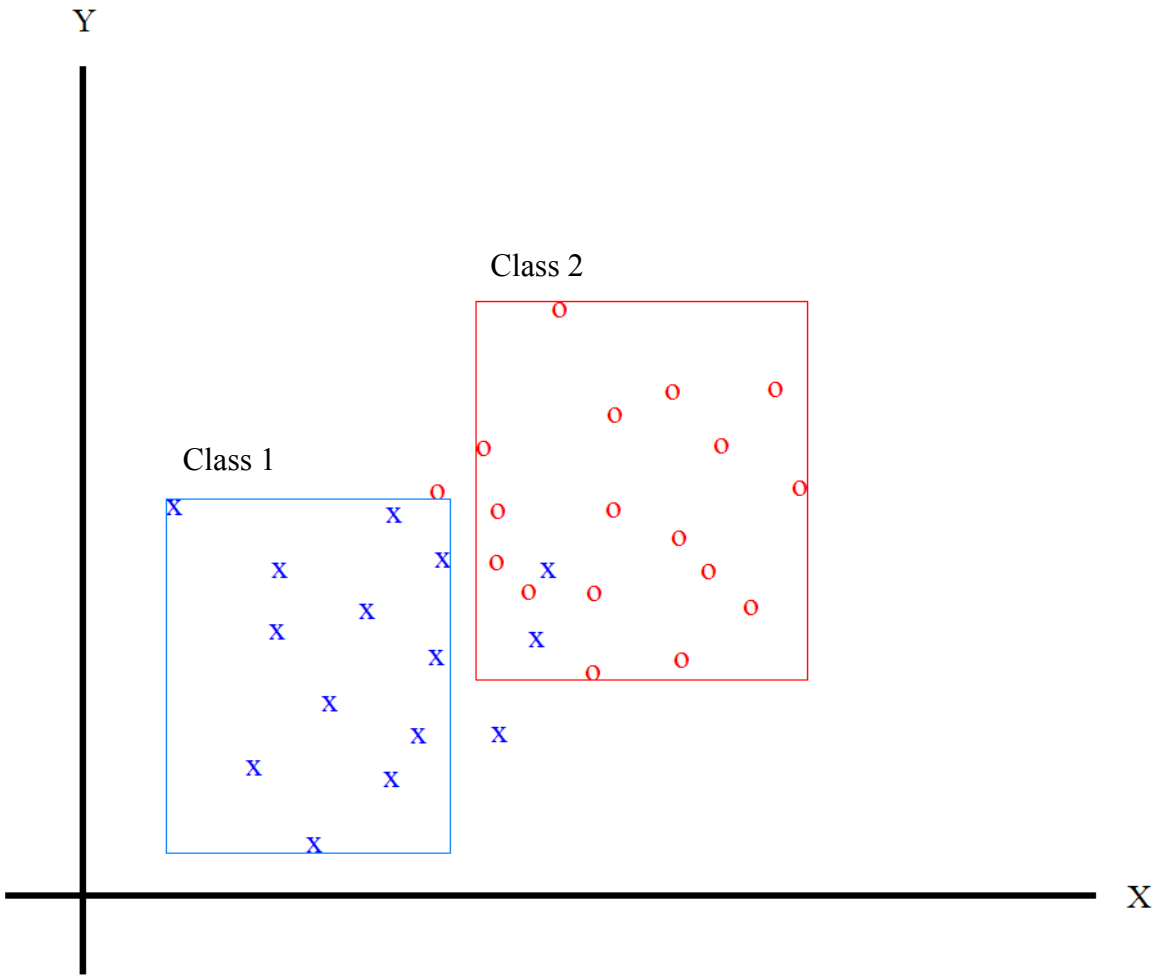


Figure 2-16. Reasonable Compromise.

With a deterministic algorithm, we can vary the dimensions of the box to get the *best* classification, i.e., highest classification accuracy (subject to a penalty for misclassification) and few (or no) points unclassified.

2.7 PSEUDOCODE, SYSTEM DIAGRAMS, AND COMPLEXITY

We now present the algorithms more formally and discuss their computational complexity.

There is negligible overhead in calculating the means and standard deviations for each of the attributes for each of the classes in the training dataset. This is done once and then used for all steps. In the following, *m*, *k*, and *n* denote the number of classes, attributes, and data points (in the training set), respectively.

Operations of multiplication, addition, subtraction, and comparison are considered equal for comparison purposes. Figures 2-17 and 2-18 show the pseudocode and the system diagram, respectively, for the classification procedure.

	Paired Planes Classification Procedure (main body)
Input	Training Set; desired accuracy A
Output	Classification Accuracies, Estimate of Overlap, Classification Parameters
	Begin { 1 Calculate training class means and standard deviations for each attribute; 2 **Algorithm 1:** (Hyperplane Algorithm); returns A_H and overlap *o* 3 **if** ($A_H \geq A$) 4 **return** A_H, *o*, Hyperplane classification parameters; 5 **else** % $A_H < A$ 6 **if** (overlap *LARGE*) 7 **Algorithm 3:** (Box Algorithm); returns A_B 8 **if** ($A_B > A_H$) 9 **return** A_B, Box classification parameters; 10 **else** % $A_B \leq A_H$ 11 **return** A_H, *o*, Hyperplane classification parameters; 12 **if** (overlap *MODERATE*) 13 **Algorithm 2:** (Margin Algorithm); returns A_M 14 **if** ($A_M > A_H$) 15 **return** A_M, Margin classification parameters; 16 **else** % $A_M \leq A_H$ 17 **Algorithm 3:** (Box Algorithm); returns A_B 18 **if** ($A_B > A_H$) 19 **return** A_B, Box classification parameters; 20 **else** % $A_B \leq A_H$ 21 **return** A_H, *o*, Hyperplane classification parameters; } **//End**

Figure 2-17. Main body of *PPCP* algorithm.

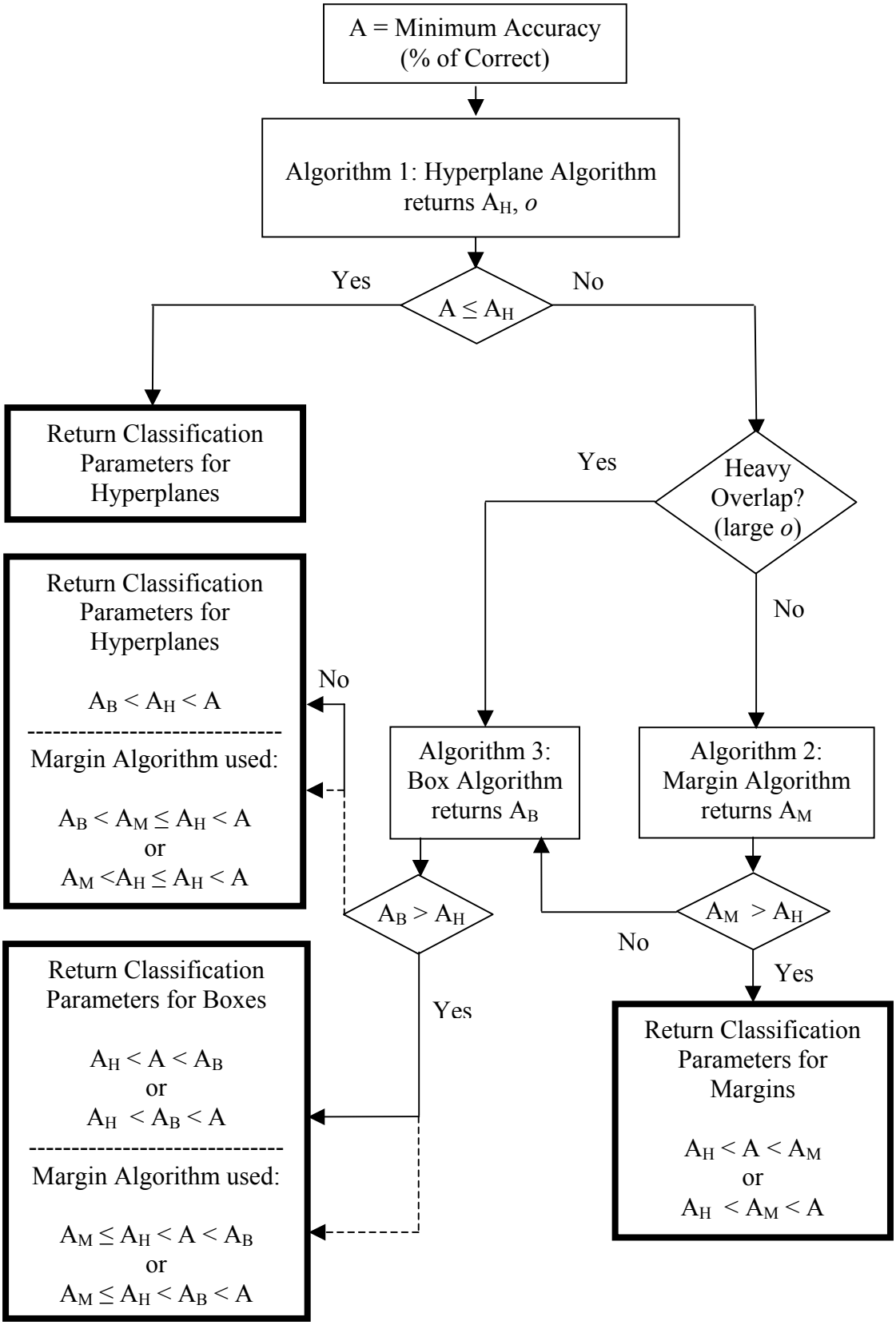


Figure 2-18. System Diagram: Complete Classification Procedure.

2.7.1 COMPLEXITY OF THE HYPERPLANE ALGORITHM

As described in section *2.4*, the hyperplanes are found by evaluating the dot product between two suitably selected vectors: the vector $\mathbf{N} = \boldsymbol{\mu}_2 - \boldsymbol{\mu}_1$, connects the means of the two classes, and vector **V** is computed for each data point **x** as $\mathbf{V} = \mathbf{N} - \mathbf{x}$. For each vector **x** on the same side of the defined hyperplane (perpendicular to **N**) as the vector $\boldsymbol{\mu}_1$, the dot product $\langle \mathbf{N}, \mathbf{V} \rangle > 0$, since $\theta = \angle(\mathbf{N}, \mathbf{V}) \in (-\frac{\pi}{2}, \frac{\pi}{2})$. Data points for which $\langle \mathbf{N}, \mathbf{V} \rangle < 0$ correspond to $|\theta| > \frac{\pi}{2}$ while those for which $\langle \mathbf{N}, \mathbf{V} \rangle = 0$ are on the hyperplane. Figure 2-19 illustrates the vectors **N**, **V** and hyperplane **H**.

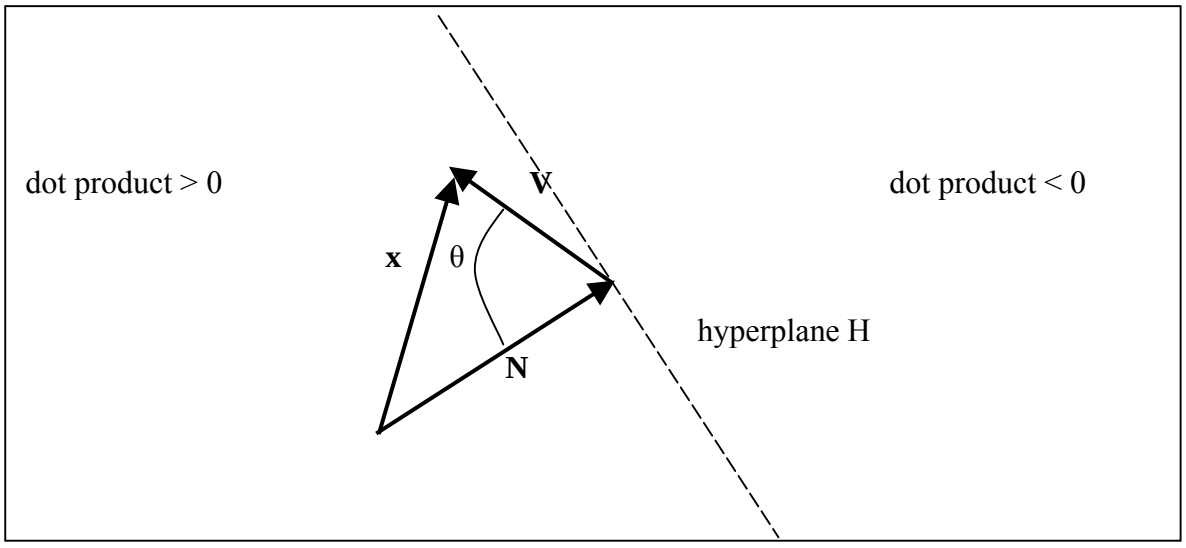


Figure 2-19. Angle Between Two Vectors.

Figures 2-20 and 2-21 show the pseudocode and system diagram, respectively, for the Hyperplane Algorithm.

Algorithm 1:	Hyperplane Algorithm (m = 2)
Input	Training Dataset, training data class means and standard deviations for each attribute, number of attributes;
Output	A_H, o, Classification Parameters
	Begin { 1 **N =** $\mu_2 - \mu_1$; 2 j = 1; 3 **while** (j > 0) 4 **N =** j**N;** 5 y_1 = 1, y_2 = 2; 6 **for** (**x, y** in the training set) 7 **V = N – x;** 8 dp = ⟨**N, V**⟩; 9 **if** (dp < 0 and **x** ∈ Class 1) 10 y_2 = 0; **11** **else** 12 **if** (dp > 0 and **x** ∈ Class 2) 13 y_1 = 0; 14 **if** (y_1 = = 1) %Class 1 points on μ_1 side of hyperplane classified w/o error. 15 **update** hyperplane **K** % **K =** j•**N** using the current value of j. 16 **if** (y_2 = = 2) %Class 2 points on μ_2 side of hyperplane classified w/o error 17 **update** hyperplane **J;** % **J =** j•**N** using the current value of j. 18 j ← j – ε } **//End**

Figure 2-20. Pseudocode: Hyperplane Algorithm.

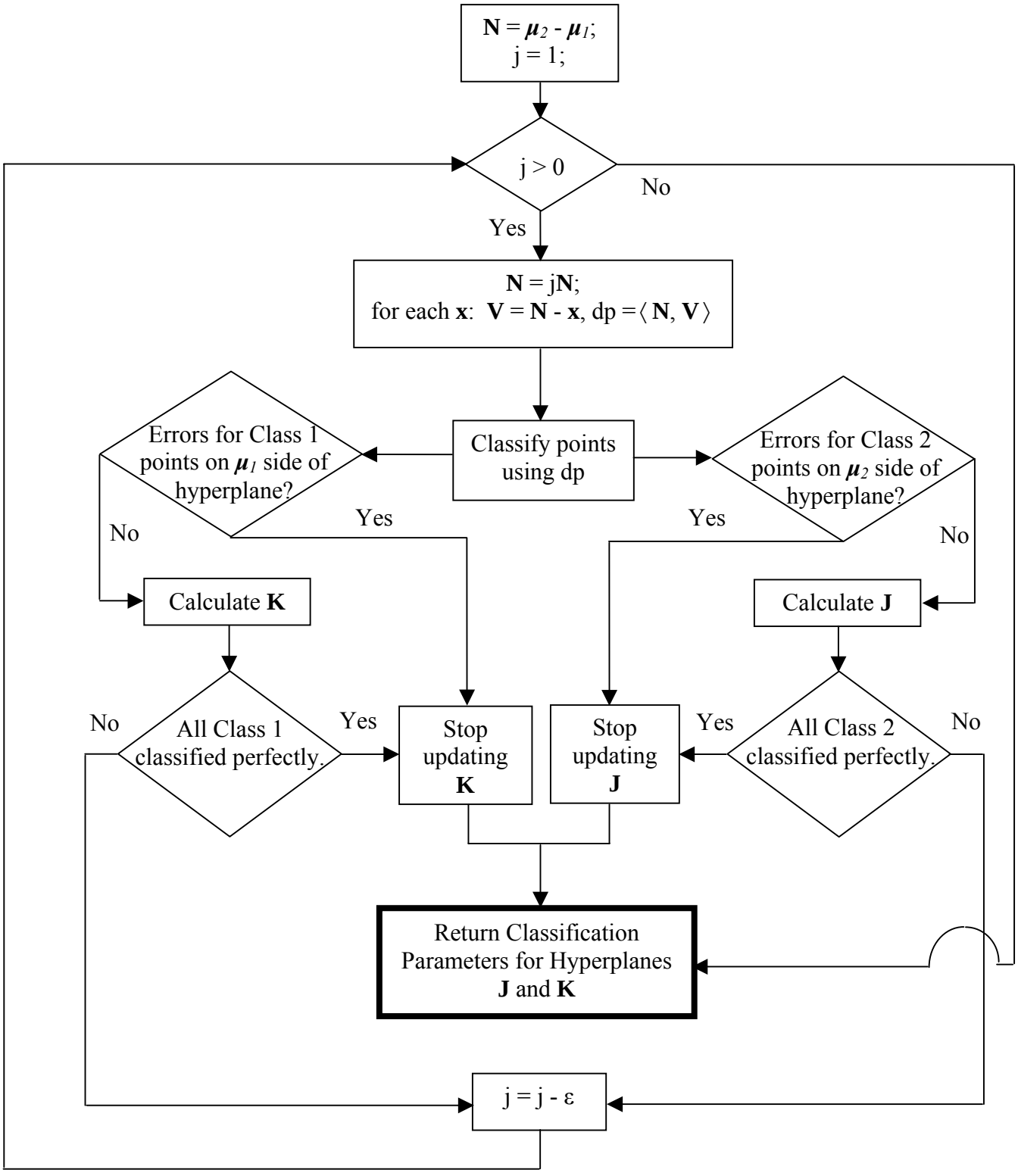


Figure 2-21. System Diagram: Hyperplane Algorithm.

2.7.1.1 Complexity of Training: Two-class Problem

In the training phase, all n points of the training set are tested for each of the j increments of a single *for loop*. A vector from each point to the vector $\mathbf{N}_j$ is formed. The dot product is found and the result compared to zero as a classification test. Each time a classification is made, correctness is checked.

There are k subtractions to form the vector from each point $\mathbf{x}$ to the vector $\mathbf{N}_j$. The classification test requires the dot product for each of the points. Since there are k attributes, there are k multiplications and $(k - 1)$ additions to form the dot product for each point. Then the dot product is compared to zero each time. Each time a classification is made, correctness is checked, so there is another comparison.

We have (k subtractions + k multiplications + $(k - 1)$ additions + 2 comparisons) = $(3k + 1)$ operations for each iteration of the loop. Thus the complexity of the training phase is linear in the number of attributes and in the size of the training set, i.e., it is $O(C_1kn)$, where the constant $C_1 = (j + 1)$ steps is determined by the stepsize j in the *for loop*.

2.7.1.2 Complexity of Testing: Two-class Problem

Using **J**, we classify each point **x** as Class *2*. If the point is not in the region for Class *2*, we use **K** to classify the point as Class *1*. If not in the region for Class *1*, it is unclassifiable (in the overlap region). As in training, the dot product is formed for each of these classification attempts. One loop is used to go through the n points.

The complexity of the testing phase is linear in the number of attributes and size of the testing set, i.e., it has an upper bound of $O(kn)$.

2.7.2 COMPLEXITY OF THE MARGIN ALGORITHM

In the classification procedure proposed here, the Margin Algorithm is *never* invoked as the first step. It is invoked after the Hyperplane Algorithm (when overlap is moderate or when A_H is too small). Other than the previously mentioned overhead for the calculation of means and standard deviations, no other work must be done in the Margin step for training. Figures 2-22 and 2-23 shows the pseudocode for the Local Margin Algorithm and the Global Margin Algorithm, respectively. The system diagram for the global version is shown in Figure 2-24.

Algorithm 2a:	Local Margin Algorithm (*m* = 2)
Input	Training Dataset, training data class means and standard deviations for each attribute, number of attributes;
Output	A_M, Classification Parameters

	Begin {
	1 **for** (i = 1 to k), %k = number of attributes
	best_accuracy$_i$ = 0;
	best_margins$_i$ = (μ^i_1, μ^i_2);
	2 **for** (η_1^k = 0 to 3, step size$_1$)
	3 **for** (η_2^k = 0 to 3, step size$_1$)
	4 **for** (i = 1 to k)
	5 current_accuracy$_i$ = 0;
	6 **calculate** current_margin$_i$ =(a_i, b_i);
	%by definition 2-1
	7 **for** (**x** in the training set)
	8 **if** (x_i < a_i)
	9 classify **x** as Class 1;
	10 **else**
	11 **if** (x_i > b_i)
	12 classify **x** as Class 2;
	13 **if** (current_accuracy$_i$ ≥ best_accuracy$_i$)
	14 **update:**
	best_accuracy$_i$ = current_accuracy$_i$;
	best_margin$_i$ = current_margin$_i$;
	%update best_margin$_i$ if
	accuracy for ith attribute improves
	15 **if** (best_accuracy = 100%)
	return A_M,
	Margin Classification Parameters;
	16 **return** A_M, Margin Classification Parameters;
	} //**End**

Figure 2-22. Pseudocode: Margin Algorithm - Local Version.

Algorithm 2b:	Global Margin Algorithm (m = 2)
Input	Training Dataset, training data class means and standard deviations for each attribute, number of attributes;
Output	A_M, Classification Parameters

	Begin {
	1 best_accuracy = 0;
	2 **for** (i = 1 to k), %k = number of attributes
	best_margin$_i$ = (μ^i_1, μ^i_2);
	3 **for** (η_1 = 0 to 3, step size$_1$)
	4 **for** (η_2 = 0 to 3, step size$_1$)
	5 current_accuracy = 0;
	6 **for** (i = 1 to k)
	7 current_margin$_i$ = (a_i, b_i)
	% by definition 2-1
	8 **for** (**x** in the training set)
	9 **if** (x_i < a_i)
	10 classify **x** as Class 1;
	11 **else**
	12 **if** (x_i > b_i)
	13 classify **x** as Class 2;
	14 **if** (current_accuracy ≥ best_accuracy)
	15 **update**:
	best_accuracy= current_accuracy,
	best_margins = current_margins;
	%update best_margin if total accuracy improves
	16 **if** (best_accuracy = 100)
	return A_M,
	Margin Classification Parameters;
	17 **return** A_M, Margin Classification Parameters;
	} **//End**

Figure 2-23. Pseudocode: Margin Algorithm – Global Version.

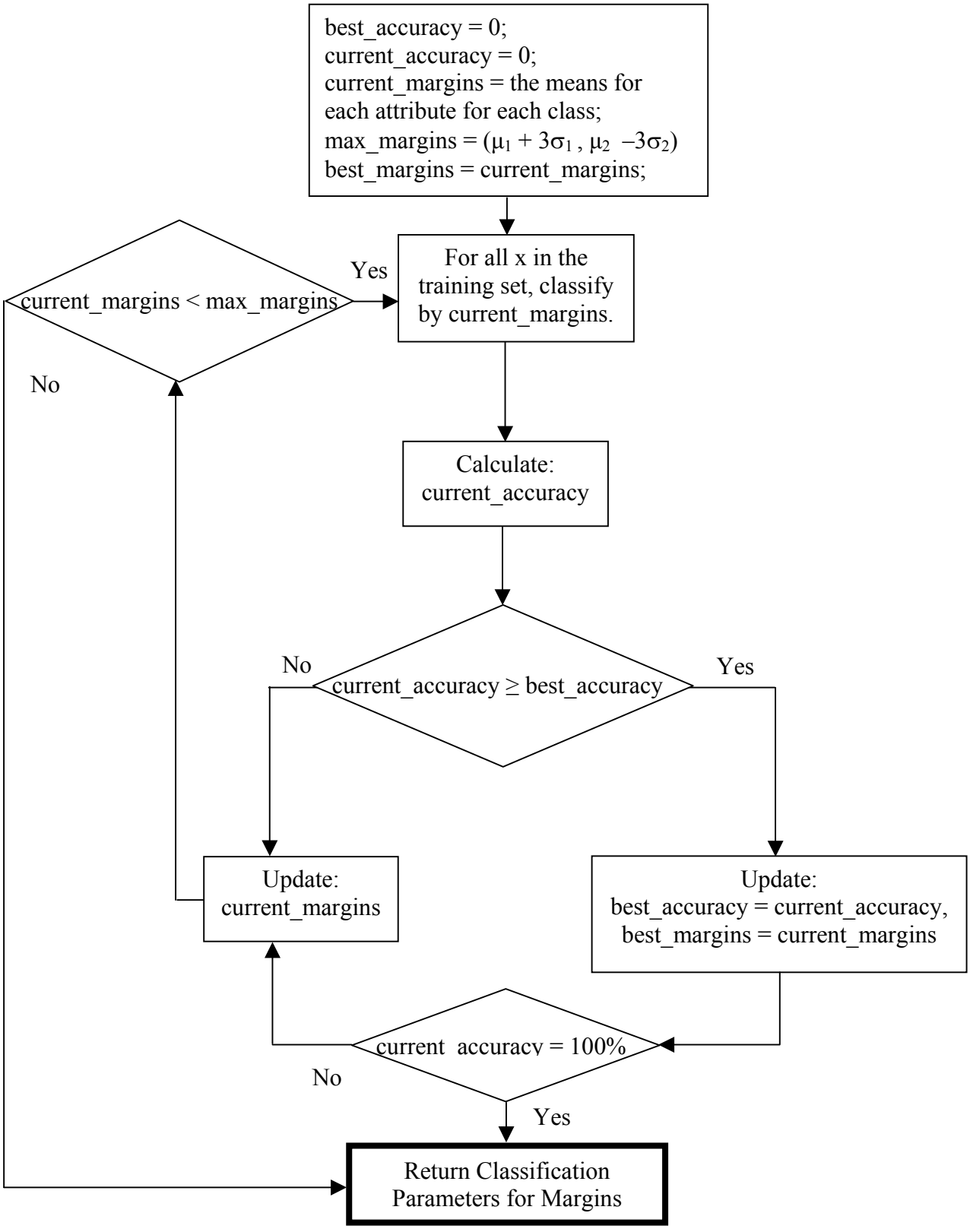


Figure 2-24. System Diagram: Margin Algorithm - Global Version.

2.7.2.1 Complexity of Training: Two-class Problem

In the training phase, *n* points are tested for each of the *j* increments of two *for loops*. For each of the *k* attributes, a *margin*, $m_k = (a_k, b_k)$, is formed between the two classes. The point is then compared to the *margin* for classification.

There are 4 operations (1 multiplication and 1 addition/subtraction to calculate each of the two values (a_k, b_k) for the k^{th} *margin*) before classifying each point. There are 4 operations (2 comparisons to the k^{th} *margin* and 2 checks for correctness) at most during classification of each point. For each point, its value for the k^{th} attribute is compared to m_k. If it is less or equal to a_k, it is classified as Class *1*. If it is greater than b_k, it is classified as Class *2*. Thus, there are 8 operations at most for a single point for each of the *k* attributes.

We have (8 operations)*k* for each iteration of the two *for loops* where the number of standard deviations is chosen for each attribute. The *j* increments in each of the two f*or loops* require $C_2 = (j + 1)^2$ steps. Thus, the complexity of the training phase is linear in the number of attributes and in the size of the training set, i.e., it is $O(C_2kn)$.

Each point is classified using the *margins* as Class *1*, Class *2*, or unclassifiable (in the overlap region).

2.7.2.2 Complexity of Testing: Two-class Problem

In the testing phase, *n* points are tested during a single *for loop*. Each point is classified (by the *margin*) using the first of the *k* attributes as Class *1* or not, by comparing to $m_k = (a_k, b_k)$. Thus, there are at most two comparisons for each test point: against a_k to classify in Class *1* and against b_k to classify in Class *2*.

Points remaining in the overlap region for this attribute *k* are then tested again using the values of the *margins* for the remaining attributes. The local version of the

Margin Algorithm requires a sort on the attributes by classification accuracy obtained in the training phase. Sorting on the k attributes requires $k\ln k$ steps.

We have (*2* comparisons)(*k)* for the single *for loop*. Thus, the complexity of the testing phase is linear in the number of attributes and in the size of the testing set, i.e., it is $O(kn)$ for the global version and $O[k(n + \ln k)]$ for the local version of the algorithm.

2.7.3 COMPLEXITY OF THE BOX ALGORITHM (CUBE)

In the classification procedure proposed here, the Box Algorithm is *never* invoked as the first step. It is invoked either after the Hyperplane Algorithm (when overlap is heavy and $A_H < A$) or after the Hyperplane Algorithm and the Margin Algorithm (when overlap is moderate and $A_M \leq A_H$). The classification boxes are controlled from quantities already calculated in the previous step.

For testing, the estimate of the overlap (from the Hyperplane step) for each class can be used to decide a *best* order. The class with the greater amount of points in the overlap is used first in classifying a generic point **x**. If one class were totally inside another, as a box inside a box, it would be likely to have more points in the overlap.

Figures 2-25 and 2-26 show the pseudocode and system diagram, respectively, for the Box Algorithm (*cube* version).

Algorithm 3:	Box Algorithm (m = 2)
Input	Training Dataset, training data class means and standard deviations for each attribute, number of attributes, number of classes;
Output	A_B, Classification Parameters

	Begin {
	1 **for** (m = 1 to 2)
	2 best_area = 0;
	best_accuracy = 0;
	$accuracy_m$ = 0;
	3 **for** (η = 0 to 3, step $size_1$)
	current_accuracy = 0;
	4 **for** (i = 1 to k)
	5 $a^k = \mu^k - \eta*\sigma^k$, $b^k = \mu^k + \eta*\sigma^k$, $s_m^k = [a^k, b^k]$
	% s_m^k = side:k^{th} edge of the box, class m
	6 **for** (**x** in the training set)
	7 **if** (**x** $\in$ box_m)
	8 Classify in class m;
	9 **calculate** current_accuracy, area,
	error; % misclassified
	10 current_accuracy =
	current_accuracy-(penalty)(error);
	11 **if** [(current_accuracy > best_accuracy)
	or (current_accuracy = best_accuracy
	& area > best_area)]
	12 **update:**
	best_accuracy = current_accuracy,
	best_sides = sides, best_area = area;
	13 $accuracy_m$ = best_accuracy;
	$best\_sides_m$ = best_sides;
	} //**End**

Figure 2-25. Pseudocode: Box Algorithm – *Cube* Version.

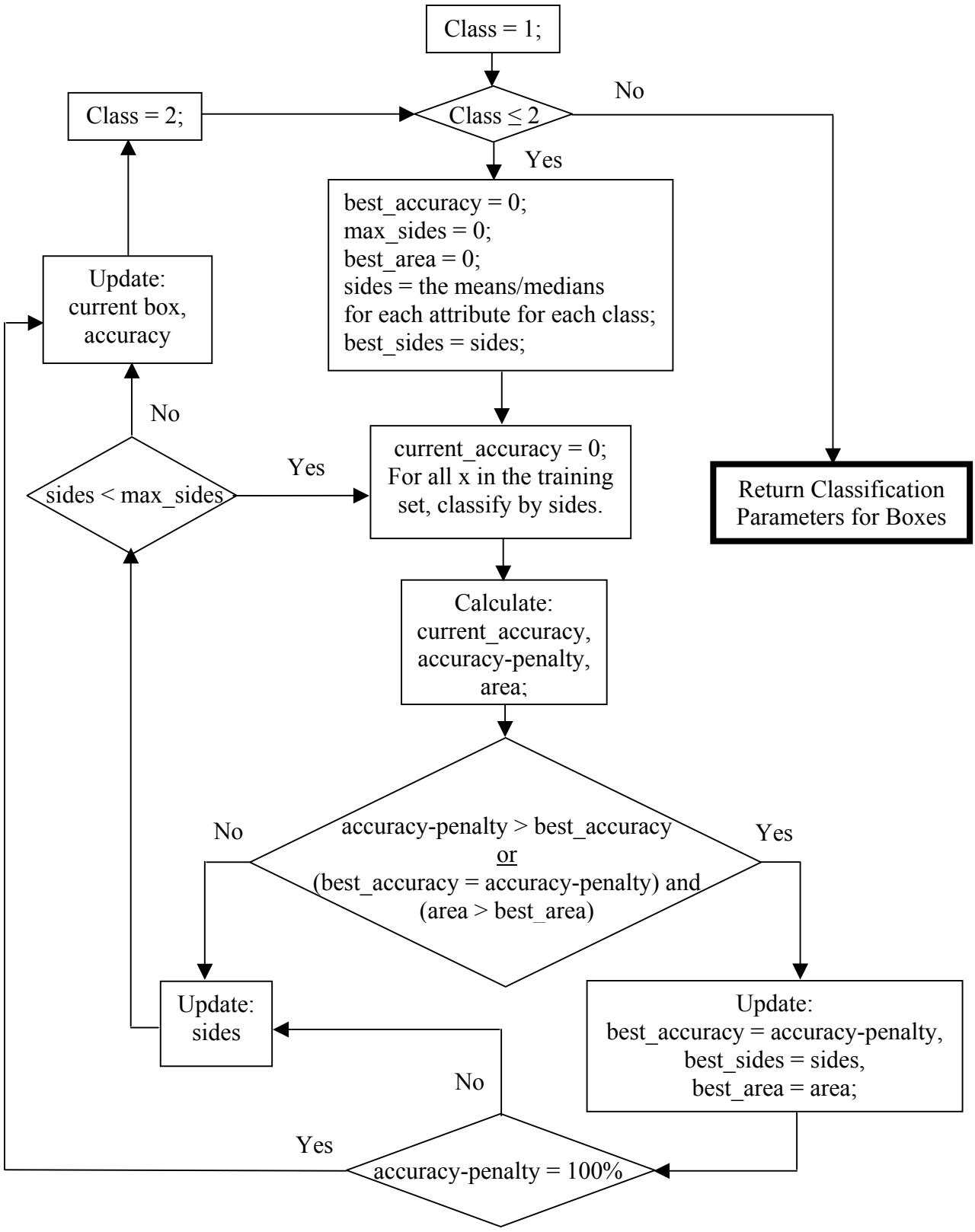


Figure 2-26. System Diagram: Box Algorithm – *Cube* Version.

2.7.3.1 Complexity of Training: Two-class Problem

In the training phase for a single class, n points are tested for each of the j increments of a single *for loop*. There are j iterations in the *for loop*, where the number of standard deviations are chosen for each attribute, thus $(j + 1)$ steps for each of the n points.

This loop is inside another *for loop* for the k attributes. For each attribute k, the formation of a $side_k = [a_k, b_k]$, requires 4 operations (1 multiplication and 1 addition/subtraction for each of the two values). After all sides have been formed, the point is then compared to the $side_k$. For each k there are 2 comparisons for purposes of classification at most: If it fails the first, the second is not done. Similarly, if it fails for any of the k sides, the remaining sides are not tested. Otherwise, it is compared for all k attributes. The same process is repeated for the remaining class.

We have $(6 \text{ operations})kn(j + 1)^2$ for each class or $2(6 \text{ operations})kn(j + 1)^2$ for two classes. Thus the complexity of the training phase is linear in the number of attributes and in the size of the training set, i.e., it is $O(C_2kn)$, where the constant C_2 is determined by the stepsize j in the *for loop*, i.e., $C_2 = (j + 1)$.

For construction of *boxes* that are not *cubes* but *symmetric rectangles*, in the training phase, n points are tested for each of the j increments of k *for loops*. Thus the constant $C_2 = (j + 1)^k$ and for large k, the complexity is extremely high consequently.

2.7.3.2 Complexity of Testing: Two-class Problem

In the testing phase, n points are tested as being in one of the 2 *boxe*s created during the training phase. A particular point is tested to see if it is in the *box* for Class *1*. If it is not in Class *1*, it is then tested to see if it is in Class *2*. Each point is classified using the *boxes* as Class *1*, Class *2*, or unclassifiable (outside both boxes).

Each point is presented to each of the k attributes sequentially for a particular class. For a particular point, its value for each of the attributes is compared to the 2 edges of the *side* for that attribute. If it is within the values for the *sides* for all attributes, it is classified as being in the *box* and therefore in that class. If it fails to fall between the two values for any attribute, it is presented for classification by the next box (class).

We have *(2 comparisons)kn* for each of the two classes. Thus the complexity of the testing phase is linear in the number of attributes and in the size of the testing set, i.e., it is $O(kn)$.

2.8 COMPLEXITY: MULTI-CLASS DATASETS

Our extension to multi-class problems is by conversion to $(m - 1)$ two-class problems. Therefore, the increase in complexity is on the order of $O(m)$.

2.9 COMPARISON TO OTHER CLASSIFIERS

We compare the Hyperplane Algorithm to Support Vector Machines. Both use the idea of a hyperplane as a decision surface, are insensitive to overtraining, and generalize well. The Hyperplane Algorithm is of complexity $O(C_1kn)$, where C_1 is determined by the stepsize j in the *for loops* of the training phase, while SVMs are of complexity $O(n^3)$ [11].

The Margin Algorithm and C4.5-type decision trees overcome *the curse of dimensionality* to some degree, as do all univariate decision trees, because only one or a few attributes are used to classify a point. C4.5-type trees require a step of discretizing continuous data, while the Margin Algorithm does not. The Margin Algorithm is of complexity $O(C_2kn)$ during the training phase, where C_2 is determined by the stepsize j in

the *for loops* of the training phase, and of complexity $O(kn)$ during the testing phase for the global version and $O[k(n + \ln k)]$ for the local version of the algorithm. By contrast, C4.5 is of total complexity $O(kn \ln n) + O[n(\ln n)^2]$, assuming k attributes, n training instances and a tree depth of $O(\ln n)$. [18] The complexity of building the tree is the first term, while the second term sums the complexities of subtree replacement, subtree raising, and average possible redistribution of the instances at every node between its leaf and the root.

The Box Algorithm is compared to k-Nearest Neighbor because both base classification on the idea that points near each other are in the same class. The Box Algorithm is of complexity $O(C_2kn)$, where C_1 is determined by the stepsize j in the *for loops* of the training phase, while K-NNs are of complexity $O(kn + n\ln n)$ as shown by the analysis below. The complexity is primarily due to classification of points.

Analysis of Complexity for K-NN

Calculation of the distance between new point with k attributes and n instances in a dataset is of the order $O(kn)$. The sort on the n points is of the order $O(n\ln n)$. Therefore, the total complexity is of the order $O(kn + n\ln n)$.

2.10 CONCLUSIONS

Paired Planes Classification Procedure (PPCP) may actually invoke all three of the algorithms discussed, i.e., the Hyperplane Algorithm, the Margin Algorithm, and the Box Algorithm. Because of this, the complexity of the classification procedure is the can range from $O(C_1kn)$ to $O(C_3kn + kln\ k)$, where $C_3 = C_1 + C_2$ as defined in section 2.7 and the maximum complexity is determined by sum of complexities of all three algorithms. The algorithms that PPCP is compared to vary in complexity from $O(kn)$ to $O(n^3)$. PPCP has a comparable complexity at the low end and better complexity by at least an order of magnitude at the high end, while being able to effectively handle a wide range of datasets.

Chapter 3:

Experiments – Artificial Datasets

Testing was carried out initially with 2-dimensional sets in order to visually confirm the process with graphs and test the viability of the three algorithms. The sets were limited to Gaussian distributions. Later, in order to test a wider ranges of datasets types, the number of dimensions was increased to four and the types of distributions were varied. This included the Gaussian, lognormal, t (student), gamma, and beta distributions. Not only were five distributions used, but also a mixture of distributions within a dataset was used. We also avoided having both classes with the same mix of distributions. Formal, precise checks of accuracy were performed using these data sets. These are presented later in this chapter.

The objectives of this phase of the work are to test our classification procedure (Hyperplane Algorithm → Margin Algorithm → Box Algorithm) and several hypotheses:

Hypothesis 1 (Margin Algorithm): A filter using rank order of classification by *least error* (least-to-most) would work better than a rank order of classification by *highest accuracy* (best-to-worst). Our definition of *least error* excludes unclassified points and is explained in section *3.2.2*.

Hypothesis 2 (Box Algorithm): The order of classes used will change the classification accuracy.

Hypothesis 3 (Box Algorithm): A *symmetric box* will have higher classification accuracy than a *hypercube*.

The results of testing on artificial sets will determine how we proceed on real-life datasets.

3.1 TWO-DIMENSIONAL DATASETS

This study was initiated with the Margin Algorithm, and then later expanded. As such, we used Matlab to create one artificial set to test the concept. The Margin Algorithm was then applied to two real-life datasets, where it proved acceptable [23]. As we extended the classification procedure to include two additional steps (the Hyperplane Algorithm and the Box Algorithm), artificial sets were created in greater numbers and complexity.

For the artificial sets used in testing the Hyperplane Algorithm and the Box Algorithm, Matlab was used to create 1000 points for each of two classes. All artificial sets created are Gaussian distributions.

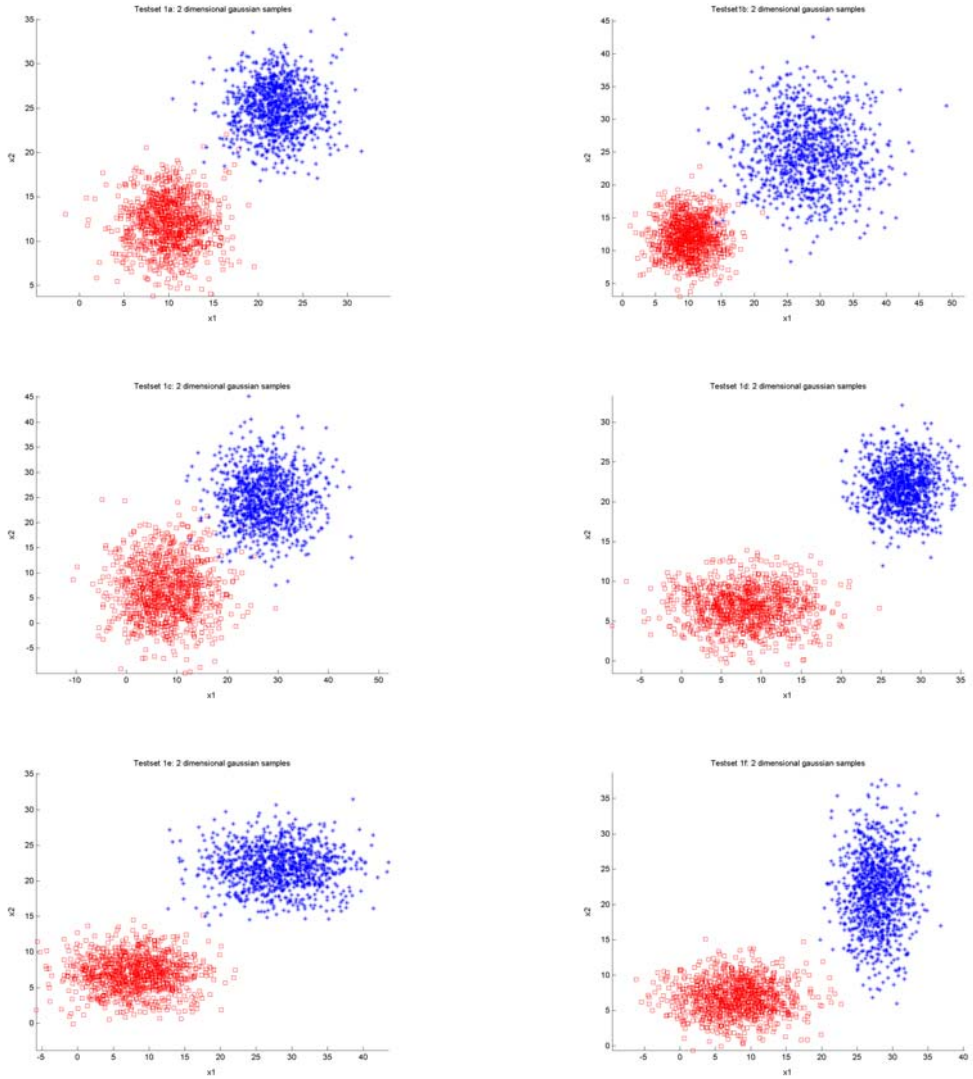
Two groups of artificial sets were considered. The first group was composed of seven cases where there was a small (less than 10%) overlap. The other group was composed of ten cases where one class was completely inside another or there was a larger (greater than 25%) overlap.

For training, we randomly selected half of each class and used the other half as a test set to cross-validate. One hundred trials were performed for each artificial set. By considering points at a distance as far as three times the maximum standard deviation of all the attributes for each class, we assured consideration all or most (greater than 97%) of the points in the class.

As stated, the intension of using these 2-dimensional, artificial datasets was to check the viability of the ideas. Visual confirmation of the viability of the concepts was

the primary goal when using 2-dimensional sets. We could construct graphs of the training sets, the test sets, and the *hyperplanes, margins, or boxes* found. For the Box Algorithm, comparisons between types of boxes (*cube, symmetric box, and asymmetric box*) and circles were made. We were more formal for this portion because we desired to not only confirm the concepts but to determine effectiveness of the various boxes [26]. This was immediately extended to two real-life datasets [27], whose results will be presented in Chapter 5. Figures 3-1 and 3-2 show the artificial sets used.

Group 1: Small Overlap



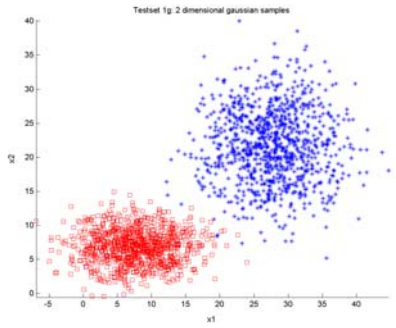
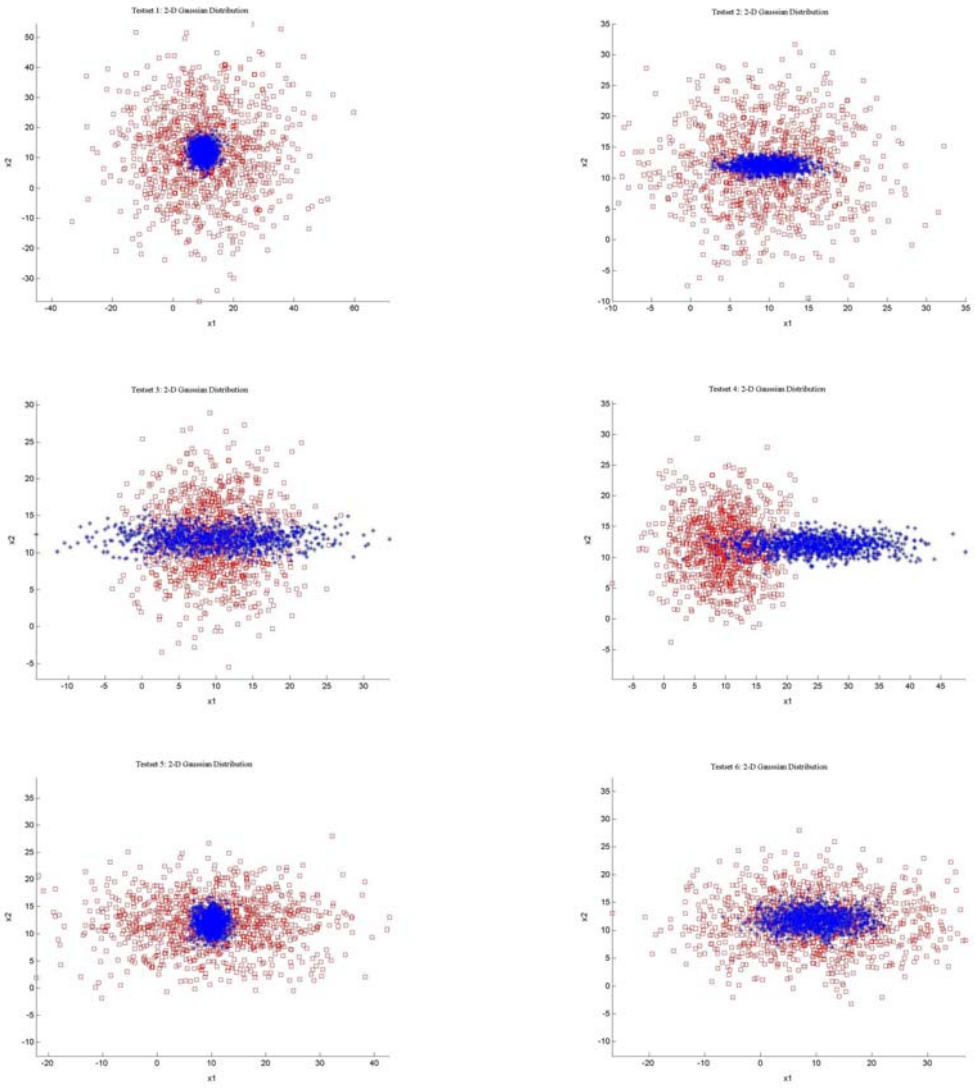


Figure 3-1. Group 1: Small Overlap Between the Two Classes.

Group 2: Extensive Overlap



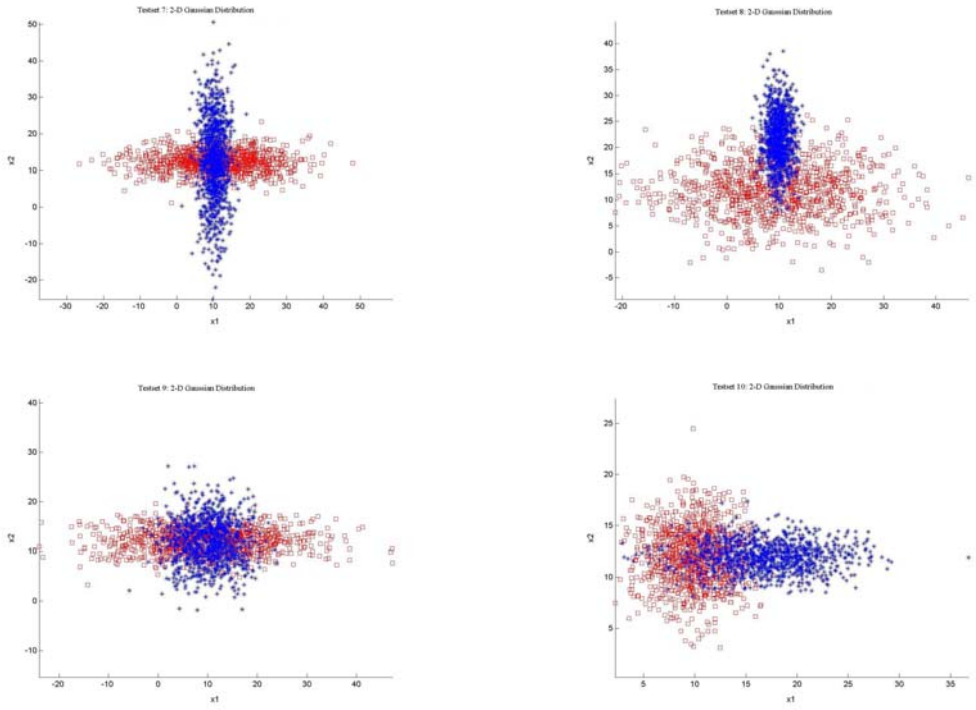


Figure 3-2. Group 2: Heavy Overlap Between the Two Classes.

3.1.1 HYPERPLANE ALGORITHM

The placement of a pair of parallel hyperplanes (as decision surfaces) was found. For a typical dataset from the first group, Figure 3-3 shows the trial set results and Figure 3-4 shows the test set when the classes are linearly separable by this method. Because there are no points in the overlap region for the trial data, the two hyperplanes are replaced by a hyperplanes midway between them and parallel to them. This is the hyperplane used in cross-validation on the test set.

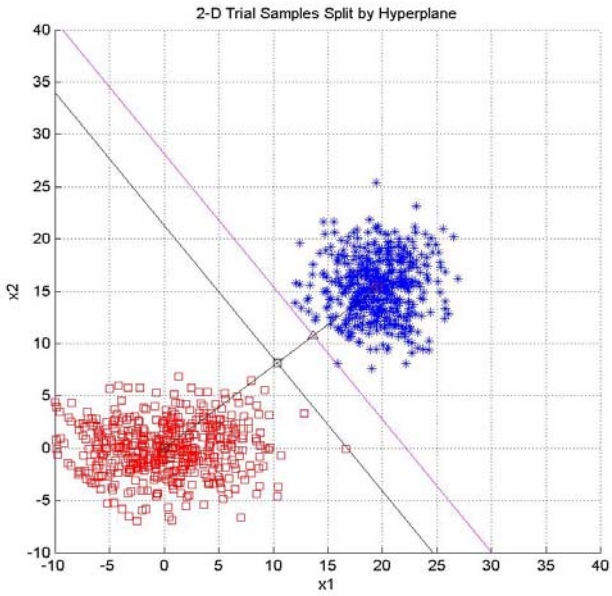


Figure 3-3. Two hyperplanes split the space. Lack of overlap allows one hyperplane.

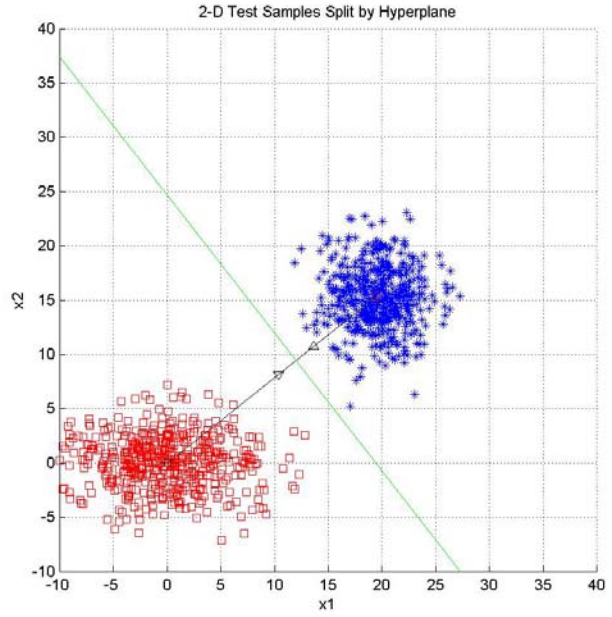


Figure 3-4. Two hyperplanes collapse to one hyperplane that is used to split the space.

For a typical dataset from the first group, Figure 3-5 shows the trial set results and Figure 3-6 shows the test set when the classes have a slight overlap. Two hyperplanes are required as decision surfaces.

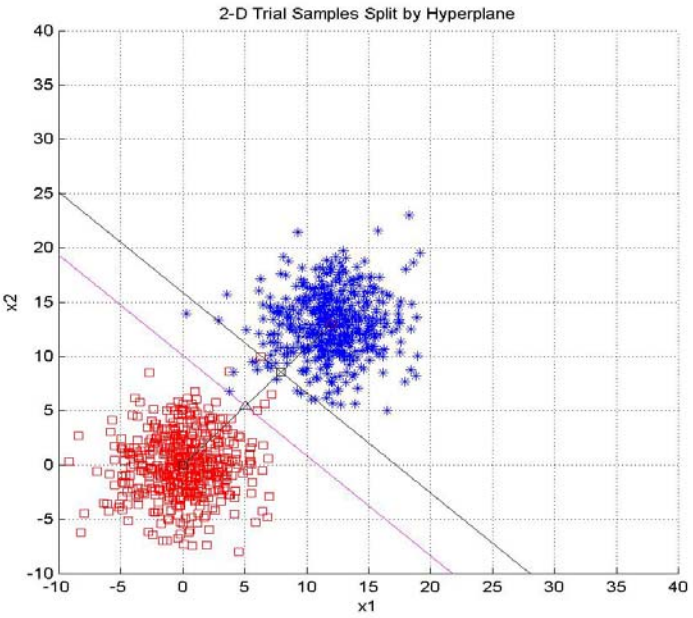


Figure 3-5. Two hyperplanes split the space. There is slight overlap between classes.

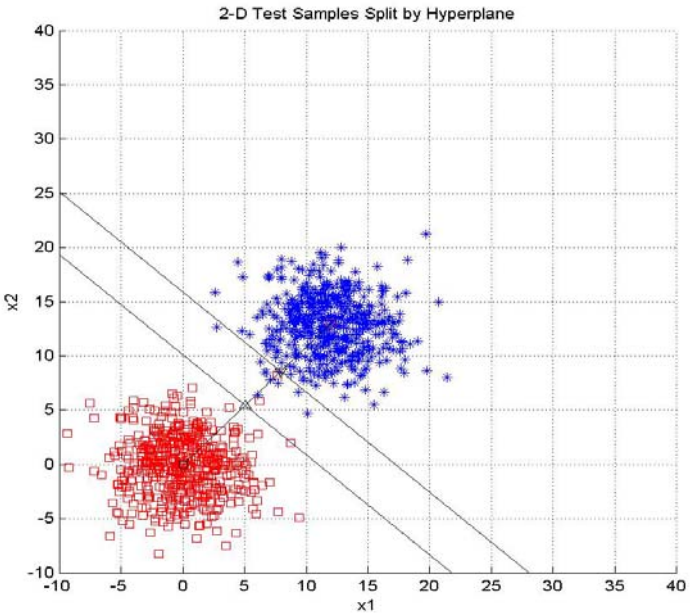


Figure 3-6. Two hyperplanes cannot collapse to one hyperplane.

For a typical dataset from the second group, Figure 3-7 shows the trial set results and Figure 3-8 shows the test set when the classes have a heavy overlap, but are not totally enmeshed or enclosed. The algorithm finds two hyperplanes. At this stage, we see that there is a large percentage of overlap and we should go to a later step of the classification procedure. When overlap is so extensive, we may jump to the third step (the Box Algorithm). However, the second step (the Margin Algorithm) may actually do better. We use the overlap estimated during the first step (the Hyperplane Algorithm) as the deciding factor. For heavier overlap (>35%), we skip the second step.

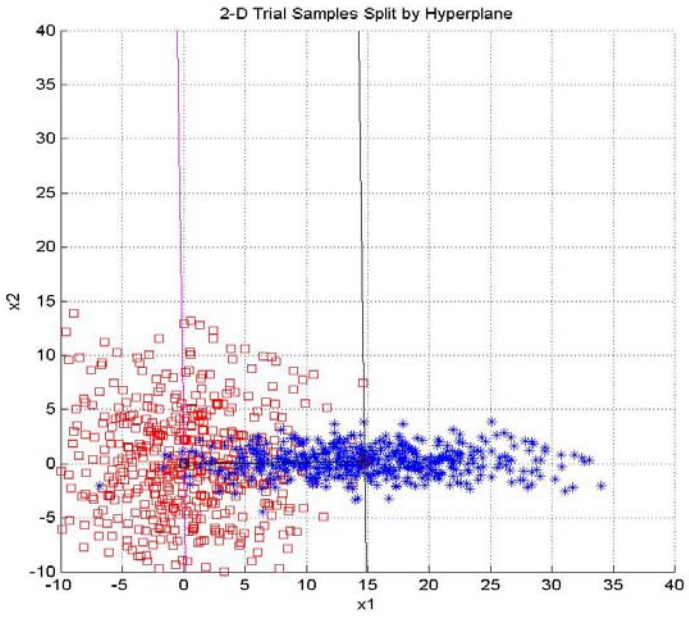


Figure 3-7. Two hyperplanes found in the training phase for the case of heavy overlap.

Note that the hyperplanes found are not quite orthogonal to the x1-axis.

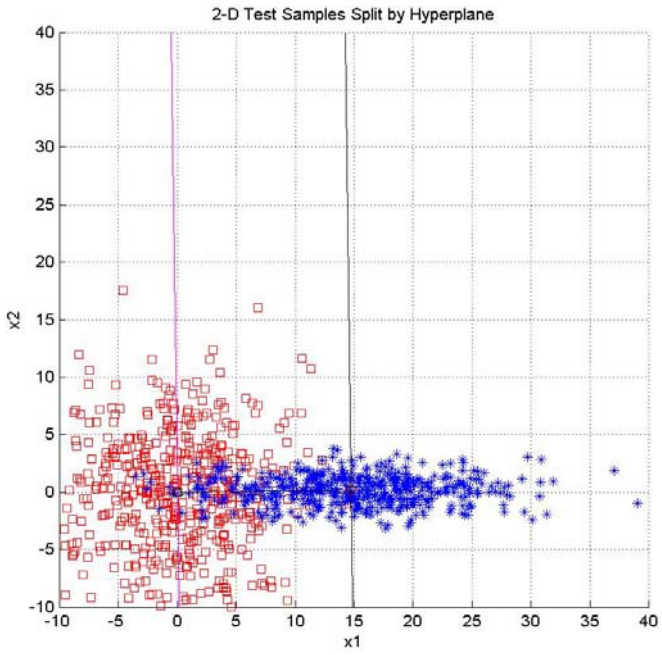


Figure 3-8. The test data is classified by the hyperplanes found in the training phase.

3.1.1.1 Results and Conclusions

For all sets in Group 1 (little overlap), the Hyperplane Algorithm was able to classify the two classes accurately and to estimate the overlap between the classes. The sets in Group 2 (heavy overlap) show the limitations of the Hyperplane Algorithm for classification as well for estimation of overlap. When the means of the two classes are the same or very close to one another, the overlap cannot be reliably estimated by this algorithm when using the current choice of starting points. However, when the hyperplanes found are at one or both of the endpoints of **N** (the vector connecting μ_1 to μ_2), we can infer that there is heavy overlap. Therefore, even when this step of the classification procedure cannot accurately estimate the overlap between classes, it still allows us to determine the next step (the Box Algorithm).

3.1.2 MARGIN ALGORITHM

A preliminary check was to create an artificial dataset by using Matlab Student Edition. A first simple version of the algorithm was done by hand and checked, using an artificial dataset (described in Table 3-1). The margin divides the feature space into regions for each of the two classes and an overlap region. For this initial version, no errors were allowed in the regions for the classes. This restriction was later eased to increase classification accuracy and robustness of the algorithm. Example 2, used to test the feasibility of *margins*, it is now shown [25].

***Example* 2:** Class *1* consists of 34 points, Class *2* consists of 232 points. Each has a normal distribution and for each attribute, $\mu_1 < \mu_2$.

Table 3-1. Example 2: Class Distribution [25].

Class Distribution			
Class	Number of Instances	μ	σ
1	34 (12.8%)	(50.21, 52.73)	(20.42, 23.12)
2	232 (87.2%)	(115.82, 111.73)	(28.94, 27.79)

In Figure 3-9 a dashed vertical line, $x = x_2$, is drawn where x_2 is the maximum coordinate for points in Class *1* and another dashed vertical line, $x = x_1$ is drawn where x_2 is the minimum x coordinate for points in Class *2*. The interval determined by these can be used to classify some of the points: those whose x coordinates fall to the left and right of the interval. However, the points with the x coordinate within the interval are not classified using their x coordinates. Since this example is just to illustrate the idea, the final position for the margin reached by the algorithm is not shown. Here the margin is simply set by using the maximum for Class *1* and the minimum for Class *2* along the x-axis.

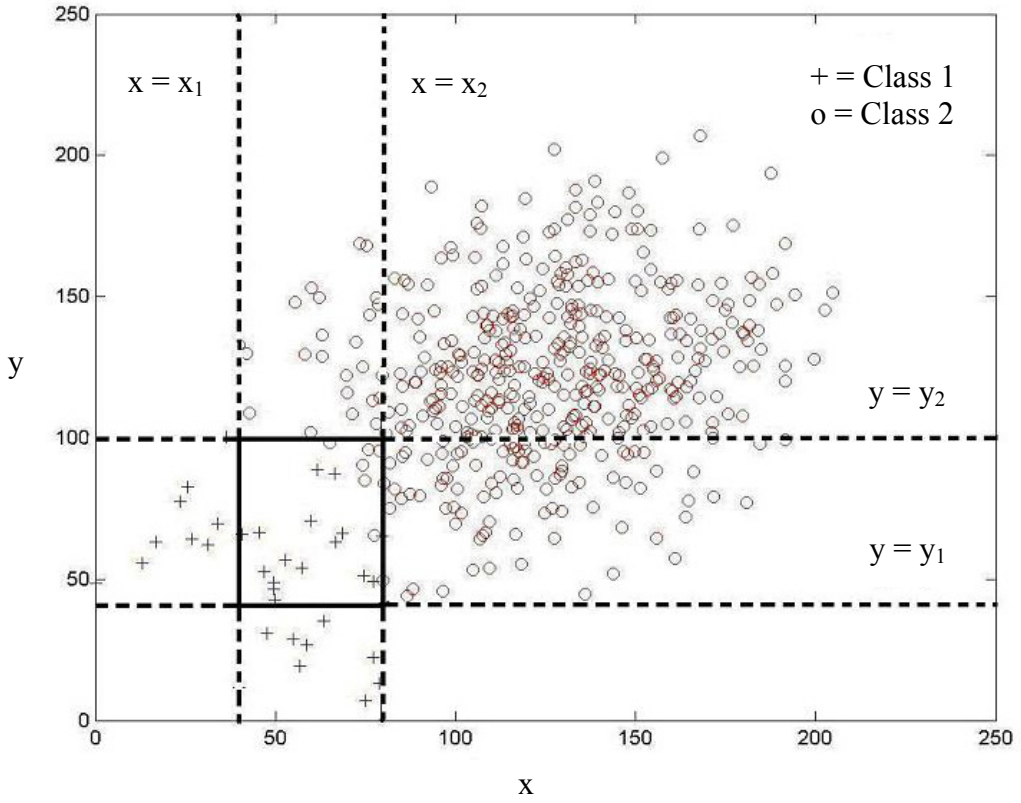


Figure 3-9. The data set for is shown with *margins*. The solid rectangle is the area where points cannot be classified.

For these points, a similar construction of a margin is done in the *y* attribute. In Figure 3-9, a dashed horizontal line, $y = y_2$, is drawn where y_2 is the maximum *y* coordinate for the points from the *y-margin* that are in Class *1* and another dashed horizontal line, $y = y_1$, is drawn, where y_1 is the minimum *y* coordinate for the points from the *y-margin* which are in Class *2*. The values y_1 and y_2 determine the *y-margin* for the attribute *y*. The points with the *y* coordinates in the *y-margin* whose *y* coordinates fall outside of it are classified as either Class *1* or Class *2*, while those whose *y* coordinates fall inside the *y-margin* cannot be classified (by this procedure). Again, for this example,

the final position reached by the algorithm is not shown. The margin is simply set by using the maximum for Class *1* and the minimum for Class *2* along the *y*-axis

3.1.2.1 Preliminary Testing: Results and Conclusions

Table 3-2 shows the combination of the results from the margins for the *x*-axis and the *y*-axis. The points inside the rectangle are not classified. All other points can be classified correctly.

Table 3-2. Example 2 results [25].

	Class 1	Class 2	Total
Correct	22	228	250
Unclassified	12	6	18
% Correct	65%	97%	93%

In this simplistic version of the algorithm, accuracy is 93.2%. The *errors* are the 18 of 266 points that remain unclassified. Even without the final positions that the algorithm attains for the margins, most points have been classified. From these results, we concluded that testing on real-life datasets would be worthwhile.

3.1.3 BOX ALGORITHM

The various boxes (*cube, symmetric box, and asymmetric box*) and a circle were compared. The *circle* was tested both with and without (*area ignored*) the maximal area heuristic, which is detailed in section *2.6*.

3.1.3.1 Results and Conclusions

Figure 3-10 shows results for the two groups of artificial sets [26], [27]. In general, the *hypercubes* did at least as well as the *circles* for all sets, whether in Group 1 or in Group 2.

There is little difference in Group 1 (small overlap). All shapes tested perform at the almost same level of classification accuracy for each set. Group 2 (heavy overlap)

shows a larger variation in classification accuracy for more than half the sets. In two sets, there is a 30%+ range in classification accuracy for the shapes. The *symmetric boxes* show a definite advantage in accuracy for Group 1 sets, i.e., those with greater overlap. The computational cost is greater than that of a *circle*. In general, it appears that:

heavy overlap $\Leftrightarrow$ rectangles' classification accuracy $\geq$ circles' classification accuracy

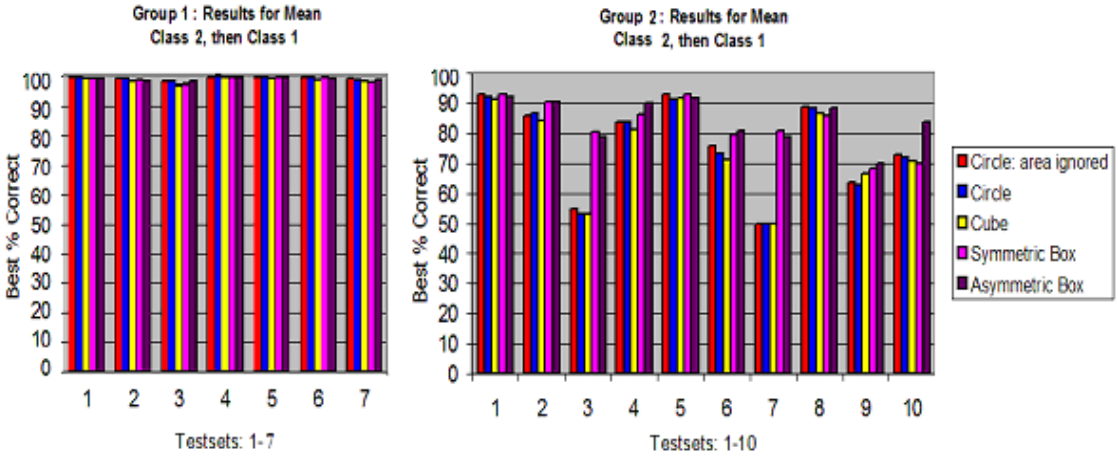


Figure 3-10. The results for all shapes, both groups of test sets.

The following hypotheses were made:

1. There would be little difference when starting with the mean or median as the center because of the symmetry of the Gaussian distribution used here, i.e., the mean and median of the distributions are almost identical. This proved true. That is why results using the median are not included in a figure.
2. The classification ability of a *circle* and a *hypercube* would be very close in value. This proved true.

3. The classification ability of a *rectangle* would be better than that of a *circle* or a *cube*. This proved true only about half the time and slightly worse a couple of times. Statistical variance could account for this being true/false on several cases.
4. The classification ability of an *asymmetric rectangle* would be better than that of a *symmetric rectangle*. This was clearly true only for set #10 (shown in the *bottom, right* of Group 2 graphs). Since *asymmetric rectangles* are extremely high in computational cost with the current implementation, use is limited to situations warranting the cost.
5. Class order (*Hypothesis 2*) would be important in for classification accuracy for moderate to heavy overlap. Though not shown here, this was true in general for cases of overlap, in particular when one class was inside another. When there was slight overlap between classes, there was little difference in classification accuracy.

3.2 FOUR-DIMENSIONAL DATASETS

For the sets, Matlab was used to create 1000 points for each class. The goal was to have a wide variety of distributions where two classes may or may not have the same distributions and where within a class, each attribute may or may not have the same distribution as other attributes. Matlab's *mvnrnd* and *gallery* functions were used, as well as several others, to do this. Matlab's inverse probability functions require symmetric, positive, semi-definite matrices to use for the covariance matrix. The appropriate function that guaranteed this was used.

We varied these choices for distributions by each of ten artificial datasets and by class within the artificial dataset. As a control, an additional dataset was created in this

manner but with 1000 points initially. By duplicating these 1000 points, two identical sets of points labeled Class *1* and the other Class *2* were formed.

Each class in a set was composed of four attributes. For each attribute in a class, the values obtained through the *mvnrnd* function were used as input in inverse functions for various distributions. This included the Gaussian, lognormal, t (student), gamma, and beta distributions. This was done separately for each class.

Only in one case was the same distribution used for all four attributes. Even then, the other class was varied in the distributions used for each attribute. Sometimes two of four dimensions were beta distribution, the third a gamma distribution, and the fourth a t-distribution, and so on.

The overlap between datasets varied from 5% to 50%. This was estimated by using the first step in the algorithm (the Hyperplane Algorithm). Each estimated overlap is viewed as a maximum. The hyperplanes used to estimate it are orthogonal to a vector that connects the means of the training classes' means. It is possible that by tilting the hyperplanes, a smaller overlap could be shown to exist. However, this estimate will be used for purposes of the full classification procedure.

The overlap of each class was also estimated. For example, in one test set, the overall overlap was estimated at 25% (500 points): Class *1*'s contribution to this was ~28% (280 points) of its examples and Class *2*'s contribution to this was ~22% (220 points) of its examples. For a different set, the overall overlap was estimated at 26%: Class *1*'s contribution to this was ~19% of its examples and Class *2*'s contribution to this was ~33% of its examples.

We randomly selected half of each class for training and used the other half of the set as a test set to cross-validate. One hundred trials were performed for each artificial set.

3.2.1 HYPERLANE ALGORITHM

Initially, the Hyperplane Algorithm was run without any error for either class allowed during the training phase. Approximations for the placement of hyperplanes were found, as were estimates of overlap *o* between classes. Then, the algorithm was run multiple times and varying amounts of errors were allowed for each class. An estimate of the *best* accuracy using this method was obtained, along with estimates for placement of the hyperplanes to do this.

Table 3-3 shows both sets of results: column 2 shows the overlap, and by inference the classification accuracy, when no errors during training were allowed. For example, when the overlap is 25%, classification accuracy = 75%: 75% = 100% – 25%.

Column 6 shows the *best* classification accuracy when some errors during training were allowed. Allowing some errors increases the accuracy by 20% or more at times. In these cases, outliers between the two classes have probably led to a larger overlap being detected. While true, we make the *apriori* decision to accept some errors in the hope of higher overall accuracy.

In Table 3-3, *best_1* and *best_2* refer to the relative length of the vector connecting the means of the two classes, as computed from the training set. Column 5 shows these values when no errors were allowed during training. Column 8 shows these values when some errors were allowed during training and we found the *best* classification accuracy.

The value 0.39 means we place a hyperplane (as a decision surface) 39% of the way along the vector **N** connecting μ_1 to μ_2 and perpendicular to **N**. It is interesting to note that the overlap between (the two identical) classes for the control set is estimated at 75%. The values of 0.00/0.00 for the separating hyperplanes in this case reflect that the algorithm can do no better than the initial placement of the hyperplanes (at the two ends of the vector connecting the means) and thus a later step (the Margin Algorithm or the Box Algorithm) in our classification procedure is appropriate.

Table 3-3. The results for all 4-dimensional test sets, including the control set.

PPCP Algortihm	Hyperplane Algorithm						
dataset #	Total: % Overlap	Class 1: % Overlap	Class 2: % Overlap	best_1/ best_2	Best % correct for 2 planes	% unclassified	best_1/ best_2
1	25	28	22	0.10/0.82	96.15 (25 errors each class)	1.9	0.39/0.46
2	26	19	33	0.19/0.77	96.4 (30 errors each class)	1.3	0.41/0.48
3	40	20	60	0.000/0.58	79.7 (150 errors Class*1*, 150 errors Class *2*)	10.0	0.60/0.90
4	8	4	12	0.00/0.64	92.1 (30 errors Class*1*, 30 errors Class *2*)	0.2	0.81/0.84
5	30	23	38	0.00/0.00	70.7 (50 errors Class*1*, 50 errors Class *2*)	10.0	0.45/0.81
6	5	4	6	0.05/0.51	95.9 (10 errors each class)	0.9	0.65/0.71
7	50	0	100	0.69/0.00	96.3 (10 errors each class)	1.0	0.73/0.79
8	11	18	4	0.26/0.66	99.3 (10 errors each class)	0.8	0.47/0.58
9	27	24	30	0.09/0.70	95.7 (20 errors each class)	0.3	0.42/0.44
10	48	49	47	0.09/0.86	94.6 (10 errors each class)	1.5	0.39/0.51
11: control	75	0	100	0.00/0.00	25 (0<n<500 errors each class)	0.0	0.00/0.00

3.2.2 MARGIN ALGORITHM

Hypothesis 2 (the order of classes) is irrelevant for the Margin Algorithm. In the context of most classification algorithms, when rank order by *least error* is used, this is typically equivalent to *highest accuracy* because most algorithms have a classification division of *accuracy* and *error*: *error* = 100% - *accuracy*. Therefore, classification using the heuristic of *best* attribute order by *highest accuracy* is equivalent to using the attributes increasing order of *least error* of classification. We term this *highest accuracy* for our comparison.

On the other hand, the Margin Algorithm has a classification division of *accuracy*, *error*, and *unclassified*. Thus, *error* ≠ 100% - *accuracy* for our algorithm, unless no points remain as *unclassified*. By *Hypothesis 1*, a rank order of classification that starts classification with the attribute with least *true* error (not including unclassified), those points as yet not classified might be classified correctly by attributes used later. The *least error* value is the accuracy of classification using the heuristic of *best* attribute order where the attributes are used in increasing order of *true* (not including unclassified) error of classification. We use this term, i.e., *least error*, for our comparison.

3.2.2.1 Results for the Global and Local Versions of the Margin Algorithm

Results for artificial datasets, numbered one through ten, and the control dataset are shown in Tables 3-4 through 3-7 for the global version and in Tables 3-8 through 3-11 for the local version of the algorithm.

Table 3-4. Margin (Global Version): Sets #1-3.

PPCP Algorithm	Margin Algorithm – global version								
dataset # -->	1			2			3		
best left margin	1.65, 0.55, 0.41, 0.27			1.71, 0.555, 0.41, 0.27			0.74, 0.57, 0.12, 0.44		
best right margin	2.49, 0.67, 0.74, 0.77			2.58, 0.67, 0.72, 0.74			11.21, 0.935, 1.30, 17.54		
Best by least error	**4, 3, 1, 2: 3.15%, 3.90%, 4.50%, 5.40%; 3/4 and 2/1 are tied**			**4, 3, 1, 2: 3.45%, 4.05%, 4.35%, 4.50%**			**1, 2, 3, 4: 1.80%, 3.90%, 4.20%, 4.65%**		
Best by highest accuracy	**3, 4, 2, 1: 82.10%, 79.20%, 71.70%, 60.40%**			**3, 4, 2, 1: 82.10%, 81.10%, 73.00%, 62.70%**			**3, 2, 1, 4: 74.20%, 44.90%, 33.90%, 30.60%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	87.91	80.74	95.09	89.74	84.26	95.21	90.45	83.59	97.31
1,2,4,3	88.42	82.60	94.24	89.09	83.15	95.03	90.48	83.58	97.37
1,3,2,4	89.44	83.49	95.40	89.20	82.32	96.07	90.08	85.73	94.43
1,3,4,2	89.97	84.90	95.03	90.05	84.49	95.62	89.64	84.95	94.34
1,4,2,3	88.08	80.93	95.23	89.56	83.40	95.72	88.33	82.39	94.26
1,4,3,2	88.20	81.22	95.17	89.60	83.21	95.99	88.30	84.01	92.59
2,1,3,4	91.61	89.75	93.46	91.27	88.29	94.24	93.94	89.15	98.73
2,1,4,3	91.11	89.12	93.10	92.27	90.42	94.12	93.75	88.93	98.57
2,3,1,4	91.60	89.42	93.78	92.50	91.41	93.59	93.66	89.29	98.03
2,3,4,1	91.32	88.71	93.93	93.10	91.86	94.35	93.67	88.96	98.38
2,4,1,3	91.75	90.25	93.26	92.50	91.12	93.88	93.51	88.48	98.54
2,4,3,1	91.12	88.17	94.07	92.28	90.20	94.35	93.48	88.85	98.12
3,1,2,4	94.62	92.75	96.50	94.99	93.84	96.13	90.55	86.21	94.89
3,1,4,2	94.84	92.93	96.74	94.73	92.90	96.56	90.90	87.46	94.33
3,2,1,4	93.78	91.93	95.64	94.83	93.87	95.80	**90.69**	**86.25**	**95.12**
3,2,4,1	93.91	91.75	96.08	94.54	93.11	95.96	90.44	85.34	95.53
3,4,1,2	95.36	93.50	97.22	95.24	93.66	96.81	89.23	83.34	95.12
3,4,2,1	**94.84**	**92.46**	**97.23**	**95.02**	**93.61**	**96.43**	90.09	85.03	95.15
4,1,2,3	93.14	91.10	95.18	93.41	90.90	95.92	86.71	79.11	94.31
4,1,3,2	93.04	90.11	95.97	93.33	90.43	96.24	86.84	80.21	93.47
4,2,1,3	92.99	90.53	95.46	93.69	91.53	95.86	87.97	82.01	93.93
4,2,3,1	93.28	90.75	95.80	94.09	92.05	96.13	88.03	81.99	94.07
4,3,1,2	94.32	92.13	96.52	94.70	93.12	96.28	86.86	82.08	91.64
4,3,2,1	94.08	91.78	96.38	94.87	93.72	96.02	87.07	82.14	92.00
maximum	**95.36**	**93.50**	**97.23**	**95.24**	**93.87**	**96.81**	**93.94**	**89.29**	**98.73**
mean	**92.03**	**88.79**	**95.27**	**92.69**	**89.87**	**95.51**	**90.19**	**84.96**	**95.43**
minimum	**87.91**	**80.74**	**93.10**	**89.09**	**82.32**	**93.59**	**86.71**	**79.11**	**91.64**
least error	**94.32**			**92.70**			**90.45**		
best accuracy	**94.84**			**95.02**			**90.69**		

Table 3-5. Margin (Global Version): Sets #4-6.

PPCP Algorithm	Margin Algorithm –global version								
dataset # -->	4			5			6		
best left margin	-0.04, 0.55, -0.06, 4.68			-4.44, 0.315, 0.36, 6.09			-4.57, 0.29, 0.35, 6.06		
best right margin	11.33, 0.71, 0.72, 8.33			11.04, 0.38, 1.64, 26.95			4.68, 0.37, 0.90, 7.78		
Best by least error	**1, 2, 3, 4: 3.20%, 4.60%, 6.20%, 6.20%; 3/4 are tied**			**2, 4, 1, 3: 3.80%, 4.00%, 4.80%, 5.60%**			**2, 4 ,3 ,1: 2.80%, 4.80%, 5.20%, 6.00%**		
Best by highest accuracy	**3, 2, 4, 1: 73.90%, 71.50%, 65.30%, 29.70%**			**2, 3, 1, 4: 98.90%, 27.00%, 23.10%, 19.70%**			**2, 4 ,3 ,1: 98.00%, 95.60%, 73.30%, 62.00%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	92.24	87.91	96.58	88.66	87.17	90.15	94.64	93.64	95.63
1,2,4,3	91.99	88.53	95.45	87.39	86.58	88.21	94.43	92.99	95.86
1,3,2,4	90.34	85.38	95.30	88.96	90.53	87.39	93.49	91.27	95.71
1,3,4,2	90.73	86.23	95.22	87.03	88.46	85.60	93.34	91.67	95.00
1,4,2,3	89.79	84.14	95.43	87.44	88.25	86.64	94.44	93.08	95.81
1,4,3,2	92.36	88.45	96.27	86.83	87.65	86.02	94.36	93.30	95.42
2,1,3,4	97.26	95.40	99.12	97.68	97.84	97.52	98.38	98.34	98.42
2,1,4,3	96.43	94.09	98.76	97.42	97.58	97.26	98.67	98.58	98.75
2,3,1,4	97.81	96.06	99.56	96.87	96.38	97.36	98.53	98.47	98.58
2,3,4,1	97.64	95.83	99.45	97.00	96.62	97.38	98.54	98.07	99.02
2,4,1,3	96.64	94.76	98.53	96.99	96.60	97.38	98.90	98.51	99.30
2,4,3,1	96.72	94.70	98.75	96.64	96.24	97.03	98.88	98.38	99.38
3,1,2,4	88.49	81.13	95.85	89.71	89.04	90.39	95.94	95.47	96.40
3,1,4,2	90.30	84.82	95.78	89.61	90.82	88.40	95.83	95.48	96.18
3,2,1,4	90.18	84.51	95.86	86.97	82.64	91.30	96.59	96.70	96.47
3,2,4,1	90.20	84.69	95.72	89.28	86.35	92.21	97.18	97.22	97.14
3,4,1,2	90.16	84.61	95.71	88.27	87.63	88.91	96.22	95.97	96.48
3,4,2,1	88.33	80.63	96.03	87.78	84.50	91.06	96.26	95.92	96.61
4,1,2,3	94.15	92.10	96.19	89.07	89.19	88.95	97.36	95.99	98.73
4,1,3,2	95.46	94.08	96.84	89.70	91.29	88.10	97.29	95.73	98.85
4,2,1,3	95.17	92.61	97.74	87.49	85.38	89.60	97.79	96.73	98.85
4,2,3,1	95.70	92.98	98.42	87.10	83.54	90.65	97.69	96.69	98.68
4,3,1,2	95.73	94.10	97.35	87.33	86.20	88.46	97.02	96.28	97.76
4,3,2,1	95.57	93.18	97.95	86.71	83.56	89.85	97.02	96.20	97.83
maximum	**97.81**	**96.06**	**99.56**	**97.68**	**97.84**	**97.52**	**98.90**	**98.58**	**99.38**
mean	**93.31**	**89.62**	**96.99**	**90.33**	**89.58**	**91.08**	**96.62**	**95.86**	**97.37**
minimum	**88.33**	**80.63**	**95.22**	**86.71**	**82.64**	**85.60**	**93.34**	**91.27**	**95.00**
least error	**92.24**			**96.99**			**98.88**		
best accuracy	**90.20**			**96.87**			**98.88**		

Table 3-6. Margin (Global Version): Sets #7-9.

PPCP Algorithm	Margin Algorithm –global version								
dataset # -->	7			8			9		
best left margin	13.75, 2.38, 5.07, -6.65			-2.01, 0.60, 0.24, 0.16			-2.01, 0.11, 0.24, 0.16		
best right margin	28.76, 42.06, 13.18, 25.33			0.16, 1.01, 0.98, 0.98			-1.59, 0.60, 1.115, 0.685		
Best by least error	**3, 4, 2, 1: 4.00%, 4.00%, 4.20%, 5.00%; 3/4 are tied**			**3, 4, 2, 1: 3.00%, 3.00%, 5.00, 5.80% and 3/4 are tied**			**2, 4, 3, 1: 0.00%, 4.60%, 4.60%, 6.80% and 3/4 are tied**		
Best by highest accuracy	**3, 4, 1, 2: 14.40%, 12.90%, 10.50%, 0.00%**			**3, 4, 1, 2: 55.20%, 33.00%, 0.00, 0.00% and 1/2 are tied**			**4, 2, 1, 3: 80.20%, 50.00%, 26.90%, 12.50%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	63.09	62.53	63.64	67.36	69.25	65.47	92.08	88.84	95.32
1,2,4,3	62.58	63.78	61.38	67.45	70.10	64.79	92.10	87.96	96.24
1,3,2,4	64.23	64.43	64.03	75.97	76.51	75.43	89.15	82.34	95.96
1,3,4,2	64.49	65.47	63.50	74.71	72.74	76.68	91.61	87.88	95.34
1,4,2,3	62.90	64.20	61.59	73.84	74.21	73.47	91.42	86.50	96.34
1,4,3,2	63.62	62.40	64.85	73.16	71.96	74.37	89.42	85.34	93.50
2,1,3,4	55.39	57.40	53.38	67.30	68.23	66.37	99.60	99.36	99.84
2,1,4,3	55.02	58.55	51.48	67.93	71.87	63.99	99.88	99.92	99.84
2,3,1,4	57.23	61.45	53.00	75.35	79.53	71.17	98.59	97.32	99.86
2,3,4,1	55.69	57.95	53.43	73.73	72.53	74.92	98.11	96.38	99.84
2,4,1,3	51.51	48.51	54.51	71.66	74.84	68.48	98.91	98.00	99.82
2,4,3,1	52.55	52.64	52.45	74.06	75.64	72.49	99.85	99.94	99.76
3,1,2,4	67.36	69.68	65.04	85.00	83.95	86.04	88.34	81.44	95.24
3,1,4,2	67.44	68.13	66.75	84.72	83.18	86.25	84.29	76.16	92.42
3,2,1,4	66.99	70.79	63.18	85.34	85.44	85.24	88.25	81.82	94.68
3,2,4,1	66.43	71.10	61.77	84.35	82.20	86.51	87.06	80.62	93.50
3,4,1,2	66.77	71.02	62.51	83.69	80.78	86.60	81.97	72.96	90.98
3,4,2,1	67.07	70.60	63.55	83.47	79.76	87.18	91.01	88.76	93.26
4,1,2,3	57.44	49.04	65.84	79.35	81.17	77.53	91.08	85.42	96.74
4,1,3,2	56.93	51.40	62.46	79.24	79.27	79.22	93.01	91.58	94.44
4,2,1,3	55.69	50.73	60.66	79.39	80.55	78.23	92.54	88.00	97.08
4,2,3,1	56.21	49.34	63.08	81.10	81.19	81.02	95.17	92.46	97.88
4,3,1,2	58.15	55.86	60.45	81.18	79.73	82.63	95.04	94.28	95.80
4,3,2,1	58.03	53.59	62.47	81.68	80.20	83.16	93.52	91.44	95.60
maximum	**67.44**	**71.10**	**66.75**	**85.34**	**85.44**	**87.18**	**99.88**	**99.94**	**99.86**
mean	**60.53**	**60.44**	**60.62**	**77.13**	**77.28**	**76.97**	**92.58**	**88.95**	**96.22**
minimum	**51.51**	**48.51**	**51.48**	**67.30**	**68.23**	**63.99**	**81.97**	**72.96**	**90.98**
least error	**67.07**			**83.47**			**99.85**		
best accuracy	**66.77**			**83.69**			**92.54**		

Table 3-7. Margin (Global Version): Set #10 and control set.

PPCP Algortihm	Margin Algorithm –global version					
dataset # -->	10			control		
best left margin	-1.38, 0.59, 0.53, -0.05			120.58, 0.17, 6.78, -0.79		
best right margin	1.14, 1.72, 2.45, 0.18			127.79, 0.19, 8.25, -0.62		
Best by least error	**2, 1, 4,3: 3.80%, 5.00%, 5.20%, 5.20%; 3/4 are tied**			**2, 3, 4, 1: 73.20%, 79.00%, 80.20%, 82.20%**		
Best by highest accuracy	**4, 3, 1, 2: 77.2%, 17.5%, 2.00%, 0.00%**			**1, 4, 3, 2: 40.00%, 39.40%, 38.40%, 37.20%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	82.94	92.87	73.01	49.28	58.26	40.30
1,2,4,3	85.10	93.26	76.93	49.38	58.23	40.52
1,3,2,4	86.62	93.21	80.04	49.36	57.95	40.78
1,3,4,2	86.42	93.93	78.92	49.53	58.00	41.05
1,4,2,3	87.61	94.27	80.94	49.33	56.57	42.08
1,4,3,2	86.54	93.49	79.59	49.36	57.11	41.62
2,1,3,4	86.16	93.46	78.87	49.46	56.81	42.10
2,1,4,3	86.14	93.28	79.00	49.45	58.13	40.76
2,3,1,4	79.83	93.08	66.58	49.37	57.44	41.29
2,3,4,1	84.54	94.70	74.39	49.37	58.21	40.53
2,4,1,3	88.35	95.29	81.40	49.48	57.62	41.35
2,4,3,1	88.73	95.12	82.33	49.70	59.40	40.00
3,1,2,4	83.69	91.85	75.54	49.45	55.79	43.11
3,1,4,2	85.43	91.90	78.97	49.32	56.50	42.13
3,2,1,4	85.38	92.14	78.62	49.40	56.85	41.96
3,2,4,1	87.75	93.31	82.19	49.26	57.11	41.42
3,4,1,2	86.69	91.41	81.96	49.47	55.74	43.20
3,4,2,1	84.39	89.66	79.11	49.36	55.56	43.17
4,1,2,3	94.66	96.60	92.72	49.37	56.13	42.62
4,1,3,2	94.71	96.73	92.70	49.54	58.62	40.47
4,2,1,3	94.82	96.91	92.73	49.52	59.26	39.78
4,2,3,1	94.61	96.41	92.80	49.41	58.75	40.08
4,3,1,2	94.62	96.61	92.63	49.34	58.02	40.67
4,3,2,1	94.84	96.96	92.73	49.62	55.45	43.78
maximum	**94.84**	**96.96**	**92.80**	**49.70**	**59.40**	**43.78**
mean	**87.94**	**94.02**	**81.86**	**49.42**	**57.40**	**41.45**
minimum	**79.83**	**89.66**	**66.58**	**49.26**	**55.45**	**39.78**
least error	**86.14**			**49.37**		
best accuracy	**94.62**			**49.36**		

Table 3-8. Margin (Local Version): Sets #1-3.

PPCP Algorithm	Margin Algorithm – local version								
dataset # -->	1			2			3		
best left margin	1.65, 0.55, 0.41, 0.27			1.71, 0.555, 0.41, 0.27			0.74, 0.57, 0.12, 0.44		
best right margin	2.49, 0.67, 0.74, 0.77			2.58, 0.67, 0.72, 0.74			11.21, 0.935, 1.30, 17.54		
Best by least error	**4, 3, 1, 2: 3.15%, 3.90%, 4.50%, 5.40%; 3/4 and 2/1 are tied**			**4, 3, 1, 2: 3.45%, 4.05%, 4.35%, 4.50%**			**1, 2, 3, 4: 1.80%, 3.90%, 4.20%, 4.65%**		
Best by highest accuracy	**3, 4, 2, 1: 82.10%, 79.20%, 71.70%, 60.40%**			**3, 4, 2, 1: 82.10%, 81.10%, 73.00%, 62.70%**			**3, 2, 1, 4: 74.20%, 44.90%, 33.90%, 30.60%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	89.05	83.38	94.72	91.27	87.42	95.12	91.12	84.76	97.48
1,2,4,3	85.97	80.72	91.22	86.67	79.10	94.24	90.16	82.58	97.74
1,3,2,4	88.60	81.42	95.78	90.95	84.68	97.22	89.23	83.60	94.86
1,3,4,2	89.86	85.54	94.18	87.75	81.96	93.54	90.09	86.38	93.80
1,4,2,3	89.44	85.14	93.74	87.68	80.56	94.80	89.24	84.62	93.86
1,4,3,2	86.56	76.22	96.90	89.25	81.56	96.94	87.15	80.46	93.84
2,1,3,4	92.72	91.98	93.46	91.34	88.94	93.74	93.44	89.56	97.32
2,1,4,3	89.79	86.30	93.28	92.08	88.76	95.40	93.59	88.94	98.24
2,3,1,4	91.68	92.28	91.08	93.73	92.96	94.50	93.94	89.28	98.60
2,3,4,1	91.30	88.34	94.26	93.36	92.32	94.40	94.10	89.80	98.40
2,4,1,3	91.93	90.76	93.10	92.99	92.30	93.68	93.93	89.10	98.76
2,4,3,1	91.51	88.26	94.76	94.02	93.94	94.10	93.60	89.10	98.10
3,1,2,4	94.33	92.26	96.40	94.73	91.92	97.54	89.12	82.32	95.92
3,1,4,2	94.46	93.04	95.88	94.53	91.90	97.16	88.08	79.92	96.24
3,2,1,4	93.63	91.54	95.72	93.86	89.96	97.76	90.83	85.80	95.86
3,2,4,1	94.18	93.72	94.64	94.83	93.28	96.38	90.60	85.76	95.44
3,4,1,2	95.75	94.26	97.24	95.75	95.58	95.92	86.09	76.30	95.88
3,4,2,1	95.54	94.08	97.00	94.49	91.74	97.24	86.92	77.98	95.86
4,1,2,3	93.48	91.66	95.30	93.55	90.94	96.16	84.20	71.88	96.52
4,1,3,2	93.30	90.66	95.94	92.36	88.36	96.36	86.24	78.82	93.66
4,2,1,3	91.62	85.36	97.88	94.52	94.84	94.20	88.21	81.50	94.92
4,2,3,1	93.11	91.02	95.20	94.58	92.62	96.54	87.36	80.20	94.52
4,3,1,2	95.03	94.62	95.44	92.70	88.74	96.66	88.02	82.96	93.08
4,3,2,1	94.90	93.08	96.72	94.89	93.88	95.90	86.59	82.04	91.14
maximum	**95.75**	**94.62**	**97.88**	**95.75**	**95.58**	**97.76**	**94.10**	**89.80**	**98.76**
mean	**91.99**	**88.99**	**94.99**	**92.58**	**89.51**	**95.65**	**89.66**	**83.49**	**95.84**
minimum	**85.97**	**76.22**	**91.08**	**86.67**	**79.10**	**93.54**	**84.20**	**71.88**	**91.14**
least error	**95.03**			**92.70**			**91.12**		
best accuracy	**95.54**			**94.49**			**90.83**		

Table 3-9. Margin (Local Version): Sets #4-6.

PPCP Algorithm	Margin Algorithm – local version								
dataset # -->	4			5			6		
best left margin	-0.04, 0.55, -0.06, 4.68			-4.44, 0.315, 0.36, 6.09			-4.57, 0.29, 0.35, 6.06		
best right margin	11.33, 0.71, 0.72, 8.33			11.04, 0.38, 1.64, 26.95			4.68, 0.37, 0.90, 7.78		
Best by least error	**1, 2, 3, 4: 3.20%, 4.60%, 6.20%, 6.20%; 3/4 are tied**			**2, 4, 1, 3: 3.80%, 4.00%, 4.80%, 5.60%**			**2, 4 ,3 ,1: 2.80%, 4.80%, 5.20%, 6.00%**		
Best by highest accuracy	**3, 2, 4, 1: 73.90%, 71.50%, 65.30%, 29.70%**			**2, 3, 1, 4: 98.90%, 27.00%, 23.10%, 19.70%**			**2, 4 ,3 ,1: 98.00%, 95.60%, 73.30%, 62.00%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	89.87	83.38	96.36	90.08	90.84	89.32	93.84	92.82	94.86
1,2,4,3	87.98	79.54	96.42	89.35	89.44	89.26	92.91	89.52	96.30
1,3,2,4	89.20	82.84	95.56	84.52	83.20	85.84	93.75	91.88	95.62
1,3,4,2	92.16	89.24	95.08	90.61	91.34	89.88	90.64	84.00	97.28
1,4,2,3	91.63	88.88	94.38	87.83	85.94	89.72	92.51	89.88	95.14
1,4,3,2	89.58	82.14	97.02	88.88	92.20	85.56	93.37	91.04	95.70
2,1,3,4	97.36	95.40	99.32	97.55	97.94	97.16	98.58	98.50	98.66
2,1,4,3	96.72	94.32	99.12	98.17	98.30	98.04	98.67	98.38	98.96
2,3,1,4	97.56	95.70	99.42	96.72	96.08	97.36	98.57	98.40	98.74
2,3,4,1	96.63	93.46	99.80	95.70	93.84	97.56	98.53	98.24	98.82
2,4,1,3	96.53	94.32	98.74	97.06	97.30	96.82	98.76	98.26	99.26
2,4,3,1	96.19	93.40	98.98	95.30	94.00	96.60	98.93	98.50	99.36
3,1,2,4	87.82	79.86	95.78	90.92	91.42	90.42	96.23	96.16	96.30
3,1,4,2	85.31	73.98	96.64	90.51	90.08	90.94	96.18	95.56	96.80
3,2,1,4	92.27	88.06	96.48	88.21	84.10	92.32	96.45	95.90	97.00
3,2,4,1	88.77	81.58	95.96	89.77	87.84	91.70	96.68	96.70	96.66
3,4,1,2	93.11	90.60	95.62	85.46	82.16	88.76	96.36	96.24	96.48
3,4,2,1	90.00	84.08	95.92	91.27	89.80	92.74	95.90	95.00	96.80
4,1,2,3	93.58	90.10	97.06	90.43	90.74	90.12	97.44	96.68	98.20
4,1,3,2	94.66	91.64	97.68	92.18	93.28	91.08	96.88	95.20	98.56
4,2,1,3	94.55	91.74	97.36	85.70	85.06	86.34	97.79	97.10	98.48
4,2,3,1	95.68	92.86	98.50	89.90	87.28	92.52	97.82	96.80	98.84
4,3,1,2	95.99	93.90	98.08	89.77	87.90	91.64	96.92	96.18	97.66
4,3,2,1	95.31	92.52	98.10	89.79	86.32	93.26	96.59	95.18	98.00
maximum	**97.56**	**95.70**	**99.80**	**98.17**	**98.30**	**98.04**	**98.93**	**98.50**	**99.36**
mean	**92.85**	**88.48**	**97.22**	**91.07**	**90.27**	**91.87**	**96.26**	**95.09**	**97.44**
minimum	**85.31**	**73.98**	**94.38**	**84.52**	**82.16**	**85.56**	**90.64**	**84.00**	**94.86**
least error	**89.87**			**97.06**			**98.93**		
best accuracy	**88.77**			**96.72**			**98.93**		

Table 3-10. Margin (Local Version): Sets #7-9.

PPCP Algorithm	Margin Algorithm – local version								
dataset # -->	7			8			9		
best left margin	13.75, 2.38, 5.07, -6.65			-2.01, 0.60, 0.24, 0.16			-2.01, 0.11, 0.24, 0.16		
best right margin	28.76, 42.06, 13.18, 25.33			0.16, 1.01, 0.98, 0.98			-1.59, 0.60, 1.115, 0.685		
Best by least error	**3, 4, 2, 1: 4.00%, 4.00%, 4.20%, 5.00%; 3/4 are tied**			**3, 4, 2, 1: 3.00%, 3.00%, 5.00, 5.80% and 3/4 are tied**			**2, 4, 3, 1: 0.00%, 4.60%, 4.60%, 6.80% and 3/4 are tied**		
Best by highest accuracy	**3, 4, 1, 2: 14.40%, 12.90%, 10.50%, 0.00%**			**3, 4, 1, 2: 55.20%, 33.00%, 0.00, 0.00% and 1/2 are tied**			**4, 2, 1, 3: 80.20%, 50.00%, 26.90%, 12.50%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	63.67	59.24	68.10	67.94	67.32	68.56	92.08	88.84	95.32
1,2,4,3	62.67	64.62	60.72	67.76	75.00	60.52	92.10	87.96	96.24
1,3,2,4	64.40	64.46	64.34	76.85	79.82	73.88	89.15	82.34	95.96
1,3,4,2	65.29	64.62	65.96	76.26	76.60	75.92	91.61	87.88	95.34
1,4,2,3	63.09	71.90	54.28	75.04	76.82	73.26	91.42	86.50	96.34
1,4,3,2	63.24	60.38	66.10	72.02	68.86	75.18	89.42	85.34	93.50
2,1,3,4	56.62	59.12	54.12	69.19	75.04	63.34	99.60	99.36	99.84
2,1,4,3	52.85	49.26	56.44	67.85	77.08	58.62	99.88	99.92	99.84
2,3,1,4	57.46	57.74	57.18	73.93	73.06	74.80	98.59	97.32	99.86
2,3,4,1	55.15	56.76	53.54	72.99	71.78	74.20	98.11	96.38	99.84
2,4,1,3	50.82	50.74	50.90	70.73	68.86	72.60	98.91	98.00	99.82
2,4,3,1	53.12	56.40	49.84	74.02	78.38	69.66	99.85	99.94	99.76
3,1,2,4	66.82	67.46	66.18	85.19	84.18	86.20	88.34	81.44	95.24
3,1,4,2	66.96	70.52	63.40	85.40	85.34	85.46	84.29	76.16	92.42
3,2,1,4	66.45	73.16	59.74	84.51	82.72	86.30	88.25	81.82	94.68
3,2,4,1	66.76	70.10	63.42	86.07	88.88	83.26	87.06	80.62	93.50
3,4,1,2	67.28	70.70	63.86	84.55	82.06	87.04	81.97	72.96	90.98
3,4,2,1	67.68	72.54	62.82	81.89	76.24	87.54	91.01	88.76	93.26
4,1,2,3	58.34	46.86	69.82	80.37	84.36	76.38	91.08	85.42	96.74
4,1,3,2	58.23	48.70	67.76	78.82	75.46	82.18	93.01	91.58	94.44
4,2,1,3	54.91	51.82	58.00	79.93	80.42	79.44	92.54	88.00	97.08
4,2,3,1	57.48	48.86	66.10	82.07	85.16	78.98	95.17	92.46	97.88
4,3,1,2	57.81	67.74	47.88	82.47	81.40	83.54	95.04	94.28	95.80
4,3,2,1	57.63	63.20	52.06	81.78	77.42	86.14	93.52	91.44	95.60
maximum	**67.68**	**73.16**	**69.82**	**86.07**	**88.88**	**87.54**	**99.88**	**99.94**	**99.86**
mean	**60.61**	**61.12**	**60.11**	**77.40**	**78.01**	**76.79**	**92.58**	**88.95**	**96.22**
minimum	**50.82**	**46.86**	**47.88**	**67.76**	**67.32**	**58.62**	**81.97**	**72.96**	**90.98**
least error	**67.68**			**81.78**			**99.85**		
best accuracy	**67.28**			**84.55**			**92.54**		

Table 3-11. Margin (Local Version): Set #10 and control set.

PPCP Algorithm	Margin Algorithm – local version					
dataset # -->	10			control		
best left margin	-1.38, 0.59, 0.53, -0.05			120.58, 0.17, 6.78, -0.79		
best right margin	1.14, 1.72, 2.45, 0.18			127.79, 0.19, 8.25, -0.62		
Best by least error	**2, 1, 4,3: 3.80%, 5.00%, 5.20%, 5.20%; 3/4 are tied**			**2, 3, 4, 1: 73.20%, 79.00%, 80.20%, 82.20%**		
Best by highest accuracy	**4, 3, 1, 2: 77.2%, 17.5%, 2.00%, 0.00%**			**1, 4, 3, 2: 40.00%, 39.40%, 38.40%, 37.20%**		
Attribute Order	Total % Correct	Class 1 % Correct	Class 2 % Correct	Total % Correct	Class 1 % Correct	Class 2 % Correct
1,2,3,4	92.24	87.64	96.84	49.47	60.60	38.34
1,2,4,3	91.82	86.78	96.86	49.17	64.80	33.54
1,3,2,4	91.70	88.92	94.48	48.54	54.26	42.82
1,3,4,2	89.06	83.18	94.94	49.64	55.20	44.08
1,4,2,3	90.32	84.18	96.46	49.16	60.02	38.30
1,4,3,2	91.15	87.00	95.30	49.30	55.60	43.00
2,1,3,4	99.50	99.24	99.76	49.63	58.78	40.48
2,1,4,3	99.39	98.96	99.82	49.72	57.02	42.42
2,3,1,4	98.33	96.98	99.68	48.83	60.74	36.92
2,3,4,1	99.09	98.32	99.86	49.72	56.48	42.96
2,4,1,3	98.68	97.42	99.94	50.27	61.04	39.50
2,4,3,1	99.41	99.04	99.78	49.01	57.52	40.50
3,1,2,4	89.67	84.10	95.24	49.19	57.10	41.28
3,1,4,2	86.62	80.58	92.66	49.90	60.44	39.36
3,2,1,4	88.23	81.80	94.66	49.32	62.84	35.80
3,2,4,1	85.08	80.94	89.22	49.56	56.08	43.04
3,4,1,2	89.68	88.02	91.34	49.91	52.88	46.94
3,4,2,1	87.90	84.50	91.30	49.49	51.48	47.50
4,1,2,3	93.65	91.54	95.76	50.08	55.64	44.52
4,1,3,2	92.27	90.36	94.18	50.26	58.76	41.76
4,2,1,3	94.79	91.88	97.70	49.42	57.76	41.08
4,2,3,1	92.57	88.28	96.86	49.18	57.96	40.40
4,3,1,2	92.52	89.70	95.34	49.07	55.38	42.76
4,3,2,1	92.26	89.36	95.16	49.24	63.40	35.08
maximum	99.50	**99.24**	**99.94**	50.27	64.80	47.50
mean	92.75	**89.53**	**95.96**	49.46	57.99	40.93
minimum	85.08	**80.58**	**89.22**	48.54	51.48	33.54
least error	**99.39**			**49.72**		
best accuracy	**92.52**			**49.30**		

3.2.2.2 Results and Conclusions

We tested *Hypothesis 1*, that classification by rank order of *least error* will give higher classification accuracy than rank order of classification by *highest accuracy*. Results for the global version and the local version of the algorithm are shown in Figures 3-11 and 3-12, respectively. The control set (#11) shows ~50% for each class, regardless of the version of the algorithm, as it should. The results are almost an exact tie: five sets are better classified using *highest accuracy* as the filter, four sets are better classified using *least error* as the filter, and there is one tie. There seems to be little difference (with these data sets) by way of accuracy on test data regarding the choice of these two filters. While we have only a small number of data sets, we must reject *Hypothesis 1* at this point.

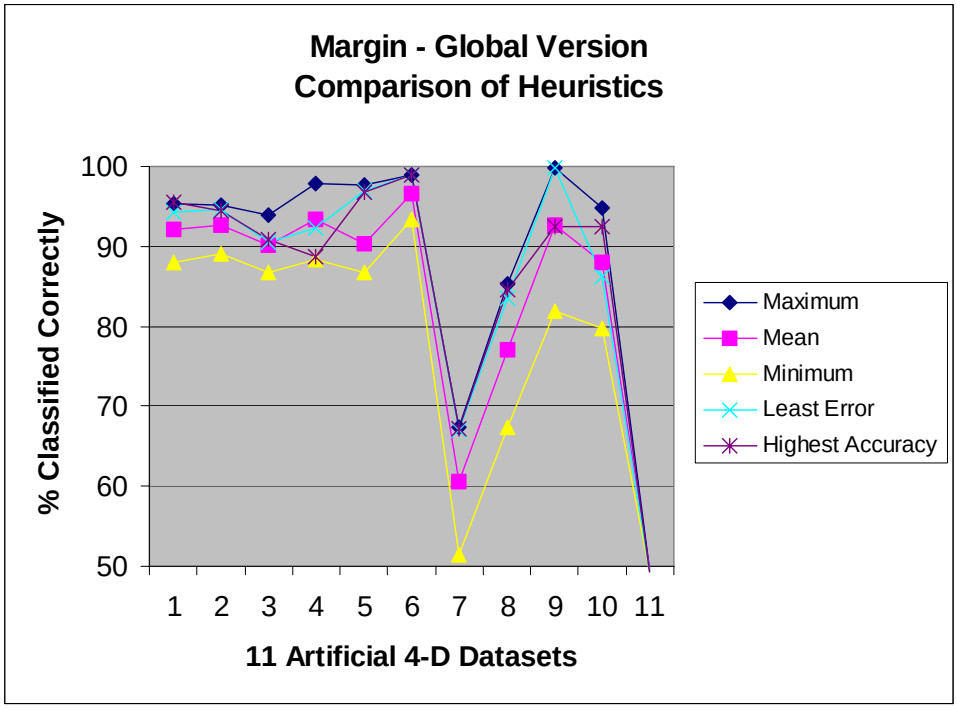


Figure3-11. Margin (Global Version): Comparison of Two Heuristics.

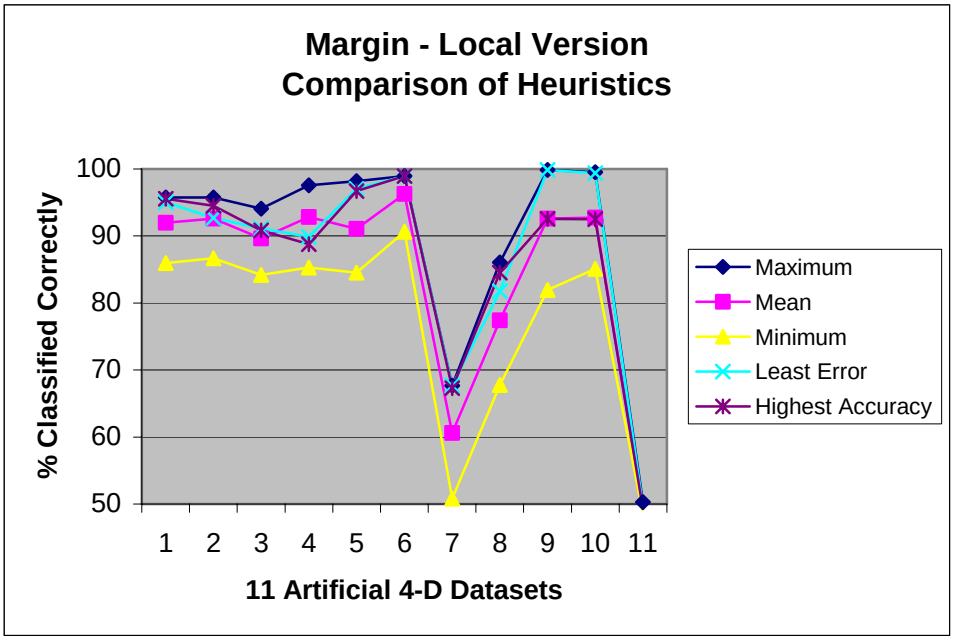


Figure 3-12. Margin (Local Version): Comparison of Two Heuristics.

3.2.3 BOX ALGORITHM

In testing with artificial, 2-dimensional sets, *Hypothesis 2* (classification accuracy by order of classes) was the same regardless of the shape used (*circles, cubes, symmetric boxes, or asymmetric boxes*). Classification accuracy by *circles* and *cubes* was similar, with *symmetric* and *asymmetric rectangles* being able to gain accuracy at higher computational cost. Testing is now extended to ten artificial sets (plus a control set) in 4-dimensions. We test two hypotheses here:

Hypothesis 2 : The order of classes used will change the classification accuracy.

Hypothesis 3 : A *symmetric box* will have higher classification accuracy than a *hypercube*.

Since an example is only classified as *inside a box* when it is within *all* dimensions of the box, *Hypothesis 1* (the order of attributes) is irrelevant..

3.2.3.1 Results and Conclusions

Tables 3-12 and 3-13 show the results of testing for *Hypothesis 2* (the class order hypothesis) and *Hypothesis 3* (the shape hypothesis), respectively.

For datasets 1, 2, 5, 7, 10, and the control set, there was little difference in the overall accuracy for the *hypercube* when *Hypothesis 2* was tested. However, the accuracies of Class *1* and Class *2* were approximately reversed. This was true for each the *symmetric rectangle* also.

For datasets 3, 4, 6, 8, and 9, there was not only a change in the overall accuracy, but also in the accuracies of each class. Again, this was true for each shape tested.

Table 3-12. Box Algorithm (Cube Version): Comparison of Class Order.

	Box Algorithm (Cube)					
	Class 1, then Class 2			Class 2, then Class 1		
dataset #	Total: % Correct	Class 1: % Correct	Class 2: % Correct	Total: % Correct	Class 1: % Correct	Class 2: % Correct
1	**55.21**	92.66	17.77	54.39	17.29	91.49
2	**57.90**	90.99	24.80	57.27	25.66	88.87
3	49.30	98.60	0.00	**65.71**	45.47	85.95
4	49.46	98.91	0.00	**74.57**	69.31	79.83
5	49.10	98.20	0.00	**49.37**	0.00	98.73
6	57.48	87.59	27.37	**71.60**	46.93	96.26
7	**49.46**	98.92	0.00	49.36	0.01	98.71
8	**53.02**	95.65	10.39	49.52	0.00	99.03
9	**68.90**	94.04	43.76	49.58	0.40	98.77
10	48.73	97.46	0.00	**49.27**	0.00	98.54
11: control	49.38	98.75	0.00	49.39	0.00	98.79

When *Hypothesis 3* was tested, seven sets showed improvement in the overall accuracy for one of the class orders and three sets showed improvement for both class orders when *symmetric boxes* were used. The improvement was usually a 3% gain. In general, whichever class gave better classification accuracy for one shape also gave better accuracy for the other shape.

Both hypotheses proved true. More work is needed to establish if a clear pattern as to class order can be determined. Use of the computationally higher *symmetric boxes* is justified when even modest gains in classification accuracy are of importance.

Table 3-13. Box Algorithm (Symmetric Rectangle Version): Comparison of Class Order

	Box Algorithm (Symmetric Rectangle)					
	Class 1, then Class 2			Class 2, then Class 1		
dataset #	Total: % Correct	Class 1: % Correct	Class 2: % Correct	Total: % Correct	Class 1: % Correct	Class 2: % Correct
1	57.18	90.88	23.48	**59.20**	31.00	87.40
2	58.85	91.06	26.64	**59.87**	29.26	90.48
3	49.20	98.40	0.00	**68.52**	45.00	92.04
4	49.24	98.48	0.00	**78.05**	68.12	87.98
5	48.87	97.74	0.00	**49.31**	0.02	98.60
6	67.69	80.08	55.30	**69.60**	79.00	60.20
7	**49.25**	98.50	0.00	49.10	0.01	98.19
8	**53.04**	95.24	10.84	49.12	0.00	98.24
9	**69.16**	94.08	44.24	52.11	10.30	93.92
10	48.54	97.08	0.00	**49.20**	0.00	98.40
11: control	49.13	98.26	0.00	49.11	0.00	98.22

3.3 CONCLUSIONS

The estimated overlap *o* found by the Hyperplane Algorithm when no errors are allowed during the training phase fails to predict which of our three steps is most appropriate to be used. This is probably due to the presence of outliers (for one or both classes) that are located between the means of the two classes. Their presence distorts the estimate. However, if we allow some errors during the training phase, the algorithm is much more robust and the appropriate steps are much more likely to be predicted.

For example, set #1 has an estimated overlap of 25% when no errors are allowed during the training phase. From this, we would choose the second step, the Margin Algorithm. If we allow some errors during training, the Hyperplane Algorithm gives 96% accuracy for a cross-validation set, which is unbeaten by other steps of the classification procedure. Both versions of the Margin Algorithm almost match this accuracy with either rank order of classification used.

The use of the overlap must therefore be tempered with the knowledge that outliers may substantially interfer with a meaningful estimate. By allowing a small amount of error during the training of parameters, this is easily overcome. When we do this, we see that no estimate of overlap exceeds 10% for set #1 and the Box Algorithm is not called as a third step.

The Box Algorithm does not classify well for set #1. In particular, whichever of the two classes is classified second is not classified with high accuracy. The sides of the box are *orthogonal* to the attribute axes. It may be that each class for this set could be successfully modeled by a box with sides *not* orthogonal to the attribute axes.

We consider the allowance of some errors during training to minimize errors during testing as a *tuning* of the algorithm. Many algorithms use a similar approach. For

instance, a decision tree algorithm will usually attain very high accuracy on training data, but a lower accuracy on testing data, i.e. previously unseen data. The classification parameters are adjusted to minimize the error on the test set and these parameters are used for future classification.

For real-life datasets in Chapter 4, we will use:

- The rank order of classification of *highest accuracy* (highest-to-lowest)
- The estimate of overlap attained for the *best* accuracy (with errors allowed during training) to determine the next step of the classification procedure.

Chapter 4:

Experiments – Real Datasets

When classifying or estimating the unknown, no one method will always work best. Certainly though, some methods work better than others do. In addition, some methods work very well in a particular circumstance. Ensemble methods that use different methods in different situations attempt to make the best of various approaches by combining them. Hybrid/ensemble classification systems which use multiple classifiers have been shown to be useful [34], [35], [36], [37], [38]. The Paired Planes Classification Procedure uses the same idea to add to performance, but the methods of the ensemble are strongly related and thus properly referred to as steps rather than different classifiers. The Paired Planes Classification Procedure performs competitively throughout, and yields transparent classifications that can be used to understand the nature of the classification.

In particular, we wish to evaluate the performance of the Paired Planes Classification Procedure on real-life datasets. As with the artificial sets in Chapter 3, the objectives here are to test our classification procedure (Hyperplane Algorithm → Margin Algorithm → Box Algorithm) and several hypotheses:

Hypothesis 1 (Margin Algorithm): A filter using rank order of classification by *highest accuracy* (best-to-worst) would improve the classification accuracy, particularly in the local version.

Hypothesis 2 (Box Algorithm): The order of classes used will change the classification accuracy.

Hypothesis 3 (Box Algorithm): A *symmetric box* will have higher classification accuracy than a *hypercube*.

Initially, we tested two datasets, each of which was a two-class problem. Algorithms classify one of the datasets, the Wisconsin Breast Cancer dataset [21], to 90%+ accuracy. This set was used to test the feasibility of the Margin Algorithm and to determine if any adjustments needed to be made. Then the Pima Indians Diabetes dataset [21] was tested. This data is *noisy* and algorithms typically classify it with 65-75% accuracy. These two datasets are representative of the range difficulty in classification and the classification accuracy attainable by most algorithms. In each of these datasets, the mean of one class is to the right of the mean of the other set for all attributes. As each new step of the classification procedure was developed, we tested it on these two datasets.

We then extended the algorithm beyond the two-class problem to the well-known Iris dataset [21]. This is a *multi-class problem* and the mean of one class is *not* to the right of the mean of the other set for all attributes. One of the three classes is linearly separable, while the other two are not.

Because there are only four attributes, the order of attributes used for classification can be evaluated. This allowed us to validate one of our heuristics, i.e., that rank order by classification ability for each attribute, is preferable for the Margin Algorithm, particularly when the local version is used. Having tested, on artificial datasets, *Hypothesis 1* (*least error* is preferable to *highest accuracy*) and rejected it, we use *highest accuracy* for rank order of classification. For the other two steps (the Hyperplane Algorithm and the Box Algorithm), all attributes are used, thus there is no order of use for the attributes, i.e., *Hypothesis 2* is irrelevant.

Each of the three steps of the classification procedure were at some time tested on the Wisconsin Breast Cancer, Pima Indians Diabetes, and Iris datasets, though not in the order specified by the overlap estimate. Initial testing was by cross-validation with half of the dataset as a test set. Each dataset was later evaluated using 5-fold cross-validation for the Hyperplane Algorithm and the Box Algorithm.

Finally, the start-to-finish classification procedure using the overlap estimate *o* is used for two additional datasets: StatLog Heart Disease [21] and Contraceptive Method Choice [21]. This allows a thorough testing of several different ideas, including using our heuristics, i.e., *rank order of classification* (by *highest accuracy*) for the local version of the Margin Algorithm, *maximal area* and a *penalty* for the Box Algorithm, and *classification by class order* for both of these; as well as our 3-step classification procedure. We used 5-fold cross-validation throughout.

In addition to using our classification procedure (Hyperplane Algorithm → Margin Algorithm → Box Algorithm) for these two datasets, we also classify by any step that would be skipped, as determined by the overlap estimate. We are thus able to state, when the overlap is used to decide the steps and their sequence, whether the procedure serves to give the best classification possible using these three algorithms.

For the Wisconsin Breast Cancer dataset, there are 16 missing attribute values. We explain in section 4.1 how we deal with these.

4.1 WISCONSIN BREAST CANCER

The Wisconsin Breast Cancer dataset is a benchmark dataset available from the Information and Computer Science Department, University of California, Irvine [21].

Of the 699 samples, 16 correspond to missing attributes. We remove these for the Margin Algorithm, which does not require their removal, unlike the Hyperplane Algorithm and the Box Algorithm, which require all attributes to have values. We use the average value for the attribute, as calculated from the training set, as a replacement for missing values when classifying by the Hyperplane or Box Algorithm. The data is composed of nine attributes, plus a patient identifier attribute and a class attribute. There are 241 positive (malignant) and 458 negative (benign) instances. Thus, about one third are positive and two thirds are negative. The class distribution and a short statistical analysis are shown in Tables 4-1 and 4-2, respectively.

This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [21]. Samples arrive periodically as Dr. Wolberg reports his clinical cases. The database therefore reflects this chronological grouping of the data. This grouping information appears immediately below, having been removed from the data itself:

Group 1: 367 instances (January 1989)

Group 2: 70 instances (October 1989)

Group 3: 31 instances (February 1990)

Group 4: 17 instances (April 1990)

Group 5: 48 instances (August 1990)

Group 6: 49 instances (Updated January 1991)

Group 7: 31 instances (June 1991)

Group 8: 86 instances (November 1991)

Total: 699 points (as of the donated database on 15 July 1992)

Number of Attributes: 10 plus the class attribute

The eleven attributes are (class attribute has been moved to last column):

1. Sample code number (id number)
2. Clump Thickness
3. Uniformity of Cell Size
4. Uniformity of Cell Shape
5. Marginal Adhesion
6. Single Epithelial Cell Size
7. Bare Nuclei
8. Bland Chromatin
9. Normal Nucleoli
10. Mitoses
11. Class: 2 for benign, 4 for malignant

Missing Attribute Values: 16

There are 16 instances in Groups 1 to 6 that contain a single missing (i.e., unavailable) attribute value, now denoted by "?" (in the downloaded dataset).

Table 4-1. Wisconsin Breast Cancer: Class Distribution.

Class Distribution – Wisconsin Breast Cancer	
Class Value	**Number of instances**
2	458 (65.5%)
4	241 (34.5%)

Table 4-2. Wisconsin Breast Cancer: Statistical Analysis.

Brief statistical analysis				
Attribute	Minimum	Maximum	Mean	Standard Deviation
Clump Thickness	1	10	4.4	2.8
Uniformity of Cell Size	1	10	3.1	3.1
Uniformity of Cell Shape	1	10	3.2	3.0
Marginal Adhesion	1	10	2.8	2.9
Single Epithelial Cell Size	1	10	3.2	2.2
Bare Nuclei	1	10	3.5	3.6
Bland Chromatin	1	10	3.4	2.4
Normal Nucleoli	1	10	2.9	3.1
Mitoses	1	10	1.6	1.7

4.1.1 HYPERPLANE ALGORITHM

By allowing some error during the training phase, we are able to attain a classification accuracy of 96.59%, with each of the classes having a 96%+ classification. This is shown in Table 4-3 and Figure 4-1.

Table 4-3. Wisconsin Breast Cancer:

various levels of error during training, classification accuracy during testing.

PPCP Algorithm	Wisconsin Breast Cancer - Hyperplane Algorithm			
Error Allowed During Training: Class 1, Class 2	0%,0%	2%,2%	3%,3%	4%,4%
Total: % Correct	80.07	93.85	96.59	96.10
Class 1: % Correct	92.15	96.45	96.58	95.66
Class 2: % Correct	56.92	88.88	96.60	96.94

The amount of error allowed during training has a significant effect on the classification ability of the hyperplanes found. In particular, the classification for Class *2* is only ~57% unless error during training is allowed. When 3% error for each class is allowed during training, classification accuracy increases for Class *1* by ~4% and for Class *2* by ~40%.

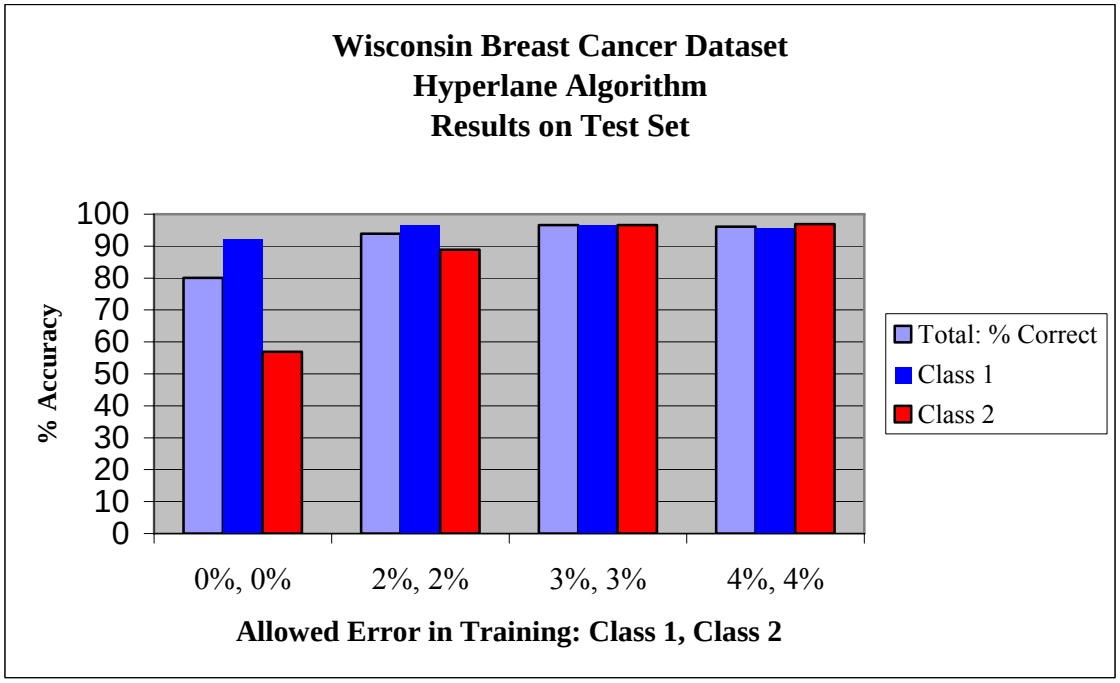


Figure 4-1. Wisconsin Breast Cancer: various levels of error during training.

By our classification procedure, we would use the classification parameters returned by this algorithm and not perform step 2 (the Margin Algorithm) or step 3 (the Box Algorithm).

4.1.2 MARGIN ALGORITHM

We present results for each, then compare the two versions.

Local Version

Table 4-4 shows the results for each version of the algorithm on ten runs of the algorithm [24]. Results over ten runs have 92.6% accuracy.

Table 4-4. Wisconsin Breast Cancer – learning constants during 10 runs of the Margin Algorithm (local version) [24].

Run #	% Correct of Test Data
1	92.3
2	93.0
3	93.0
4	93.0
5	91.5
6	91.9
7	92.1
8	92.3
9	93.4
10	93.4
Mean:	92.6

Global Version

Table 4-5 shows the learning constants found for the algorithm as well as the classification accuracy for both training and test sets during ten sample runs [25]. The average classification accuracy for training was very close to the average classification accuracy for the test sets. The learning constants indicate that on average the margins are approximately one standard deviation from the mean for each class (as computed from

the training data) and toward the mean of the other class. We can infer that there is little overlap and that the classes are *compact*, i.e., the points are near one another at the mean.

Table 4-5. Wisconsin Breast Cancer – learning constants during 10 runs of the Margin Algorithm (global version) [24].

Run #	η_A	η_B	% Correct of Training Data	% Correct of Test Data
1	2	2	93.0	94.2
2	0	0.1	94.7	95.5
3	0	0	95.6	94.6
4	0	0	96.9	95.6
5	1.9	1.7	94.7	95.1
6	1.8	2	96.9	92.5
7	0	0.3	94.3	94.1
8	0	0.3	96.5	95.5
9	2	1.6	94.3	95.3
10	1.7	1.8	93.4	94.4
Mean:	.94	.98	95.0	94.7

On a training run of the entire dataset, 25 of 444 benign cases were misclassified (5.6%), 9 of 239 malignant cases were misclassified (4.8%), and 3 of the 683 cases were not classified (0.4%), for a total of 94.6% classified accurately.

Comparison of Versions

Using the same training data, the learning constants were found for each version and then used on the same test set. Results over ten runs are shown in Table 4-6 [25]. The global version consistently classifies more accurately by ~2%.

Table 4-6. Wisconsin Breast Cancer:

comparison between versions of the Margin Algorithm [25].

Run #	% Correct of Test Data	% Correct of Test Data
	Local Version	Global Version
1	92.3	94.2
2	93.0	95.5
3	93.0	94.6
4	93.0	95.6
5	91.5	95.1
6	91.9	92.5
7	92.1	94.1
8	92.3	95.5
9	93.4	95.3
10	93.4	94.4
Mean:	92.6	94.7

4.1.3 BOX ALGORITHM

The results range from 91.13% to 94.36% average accuracy of classification for the Wisconsin Breast Cancer dataset for the initial testing (using *symmetric rectangles*). The results are averages of 100 trials with 50:50 splits of the data between training and testing [26].

In this initial testing, a *penalty*, for misclassification of points, $0 \leq penalty \leq 1$ by a step size of 0.1 was evaluated. Figure 4-2 shows the results. At $penalty = 0.3$, the classification accuracy stabilized. At $penalty > 0.6$, the classification accuracy appeared to decrease from the maximum. The highest classification accuracy appears to be when $0.2.5 < penalty < 0.7$. Similar behavior was observed for the Pima Indians Diabetes

dataset. Based on the results for these disparate datasets, a *penalty* = 0.4 was deemed reasonable to use on all other sets to be tested in this study.

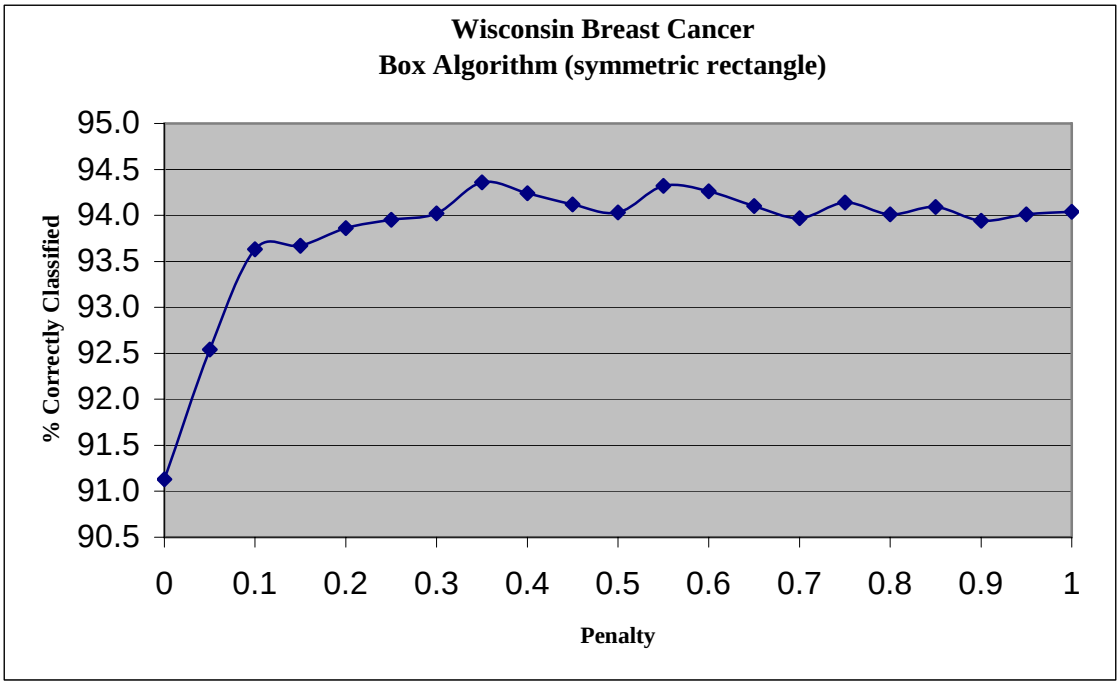


Figure 4-2. Results averaged over 100 trials for each value of the penalty tested [26].

Subsequently, with 5-fold cross-validation, the *hypercube* version had an average overall accuracy of ~88% (shown in Table 4-7 and Figure 4-3) for class order 1→2, i.e., Class *1* is used followed by Class *2*. In both orders, Class *2* was 91-92% accurate. *Hypothesis 2* is supported, i.e., not only is the overall accuracy changed by class order, but also the individual class accuracy.

Table 4-7. Wisconsin Breast Cancer (5-fold cross-validation):

Box Algorithm (cube).

PPCP Algorithm	Wisconsin Breast Cancer Box Algorithm (cube)	
Class Order	1,2	2,1
Total: % Correct	87.55	58.12
Class 1: % Correct	85.91	40.16
Class 2: % Correct	90.68	92.21

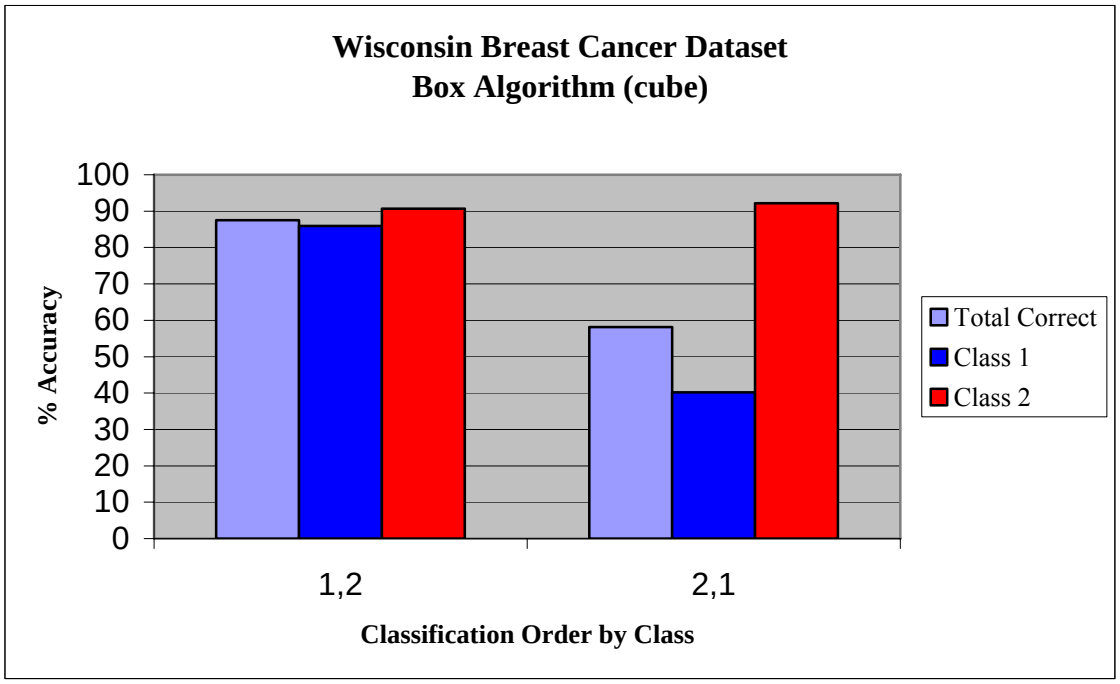


Figure 4-3. Wisconsin Breast Cancer (5-fold cross-validation):

Box Algorithm (cube).

The *symmetric rectangle* had an average total accuracy of ~92% (shown in Table 4-8 and Figure 4-4) with each of the two orders of classes used to classify. In the order of Class *1* followed by Class *2*, Class *1* was ~91% accurate and class 2 was ~94% accurate. In the order of class 2 followed by Class *1*, Class *1* was ~90% accurate and Class *2* was ~96% accurate.

Table 4-8. Wisconsin Breast Cancer (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

PPCP Algorithm	Wisconsin Breast Cancer Box Algorithm (symmetric rectangle)	
Class Order	1,2	2,1
Total: % Correct	92.38	92.14
Class 1: % Correct	91.30	90.22
Class 2: % Correct	94.44	95.83

There is a slight difference in classification accuracy overall, as well as for individual class, with the two class orders. It is not sufficient to support *Hypothesis 2*.

Hypothesis 3 is supported for this dataset: the overall classification accuracy is increased for each order, as is the classification accuracy for the individual classes. Note that Class *2* is classified with 90%+ accuracy irregardless of the class order or the shape used here. We infer that it is (to a large degree) symmetric about the mean in the original *k*-dimensional space, thus a *hypercube* can model it well. On the other hand, Class *1* does not exhibit this type of symmetry, but symmetry by each attribute axis instead. Therefore, a *symmetric rectangle* models it well.

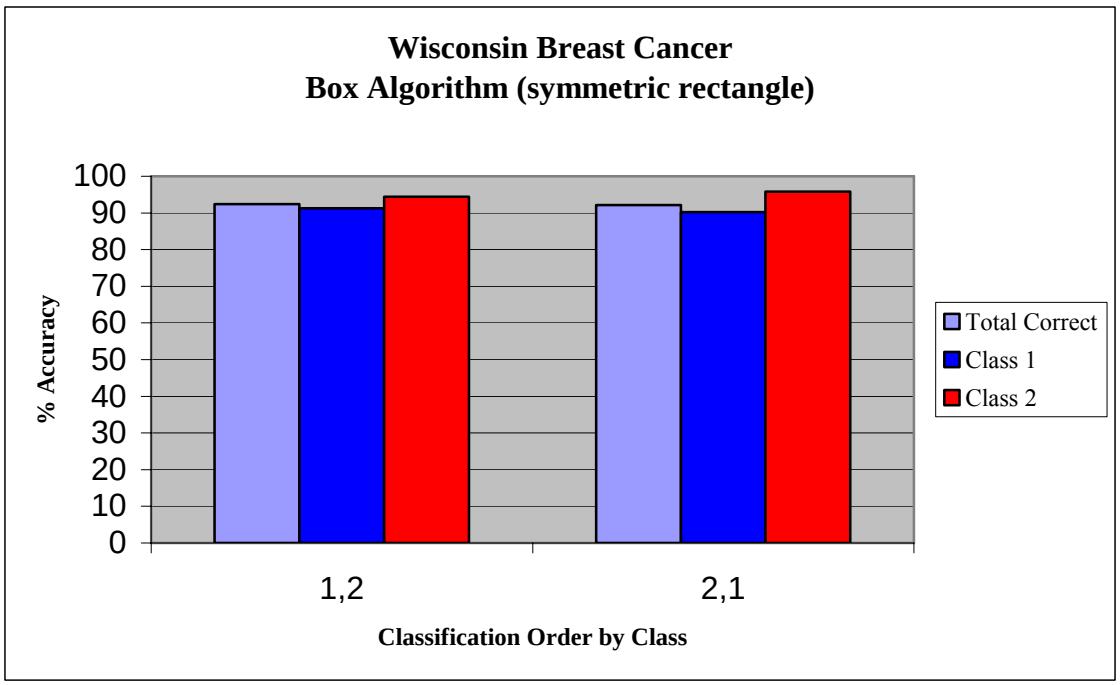


Figure 4-4. Wisconsin Breast Cancer (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

4.1.4 DISTRIBUTIONS OF THE CLASSES

It is instructive to look at a graph of the distributions of the two classes. By means of a histogram, an estimate of these distributions was created using the trial data. It is shown below in Figure 4-5. To mark the means, 0 was used for Class *1* and 1 was used for Class *2*. As the reader may recall, the Hyperplane Algorithm uses a constant times the vector connecting the two means in order to find the hyperplanes desired. It is by the Hyperplane Algorithm that a count for the histogram is obtained. The position relative to this vector is generally from 0 to 1, with the constant marking the position between the means.

We see in the case of the Wisconsin Breast Cancer dataset that while the distributions for the two classes overlap, the majority of examples are *not* in the region of

overlap. Choosing a value between 0.35 and 0.55 gives high accuracy to the classification. The Hyperplane Algorithm finds two such values as hyperplanes. The Box Algorithm can also do well when given such distributions for the two classes.

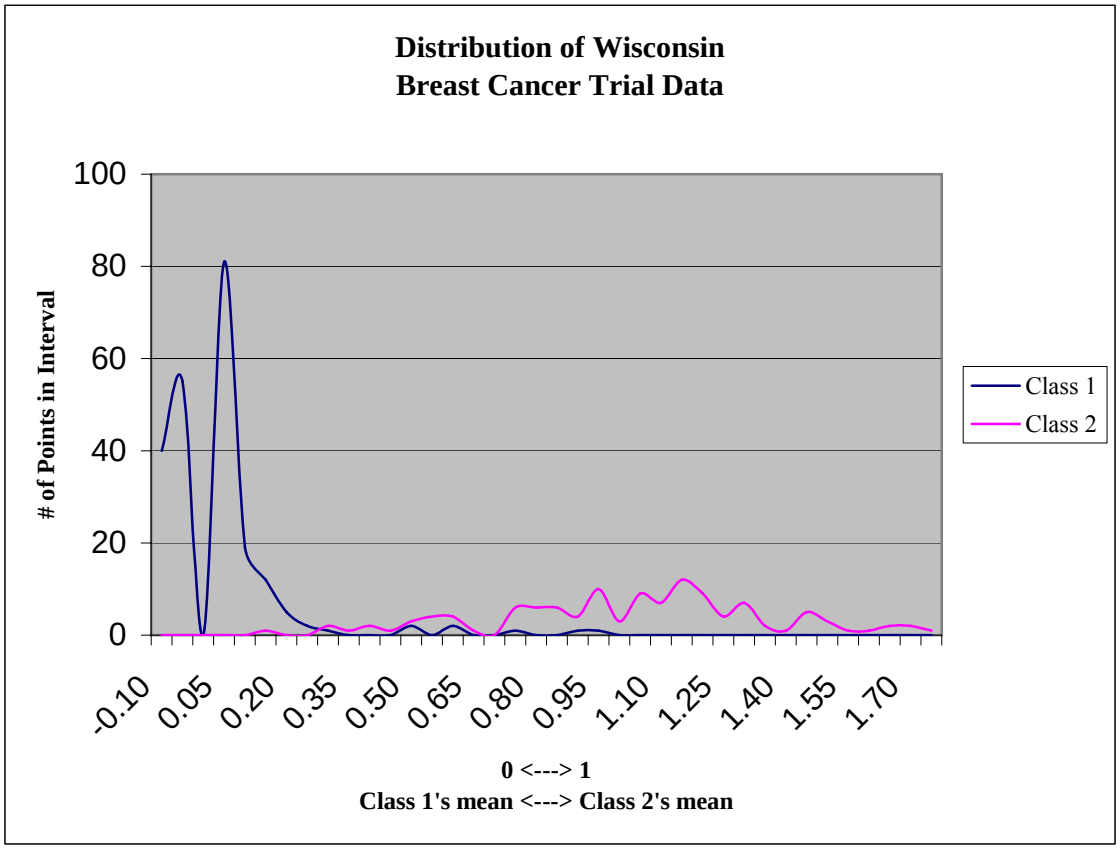


Figure 4-5. Wisconsin Breast Cancer:

Approximate Distribution Curves of the Two Classes.

Next, compare our approximate distribution curves with the densities of points normal to the separating plane obtained by Mangasarian in [10] and reproduced here in Figure 4-6 (originally labelled Figure 3). This was published in their landmark paper describing the Xcyt Image Analysis Program. The benign class on the left (*not* in the original feature space of ten attributes) in Figure 4-6 corresponds to our Class *1* on the

left (in the original feature space of nine attributes) in Figure 4-5. Each shows that the overlap is slight.

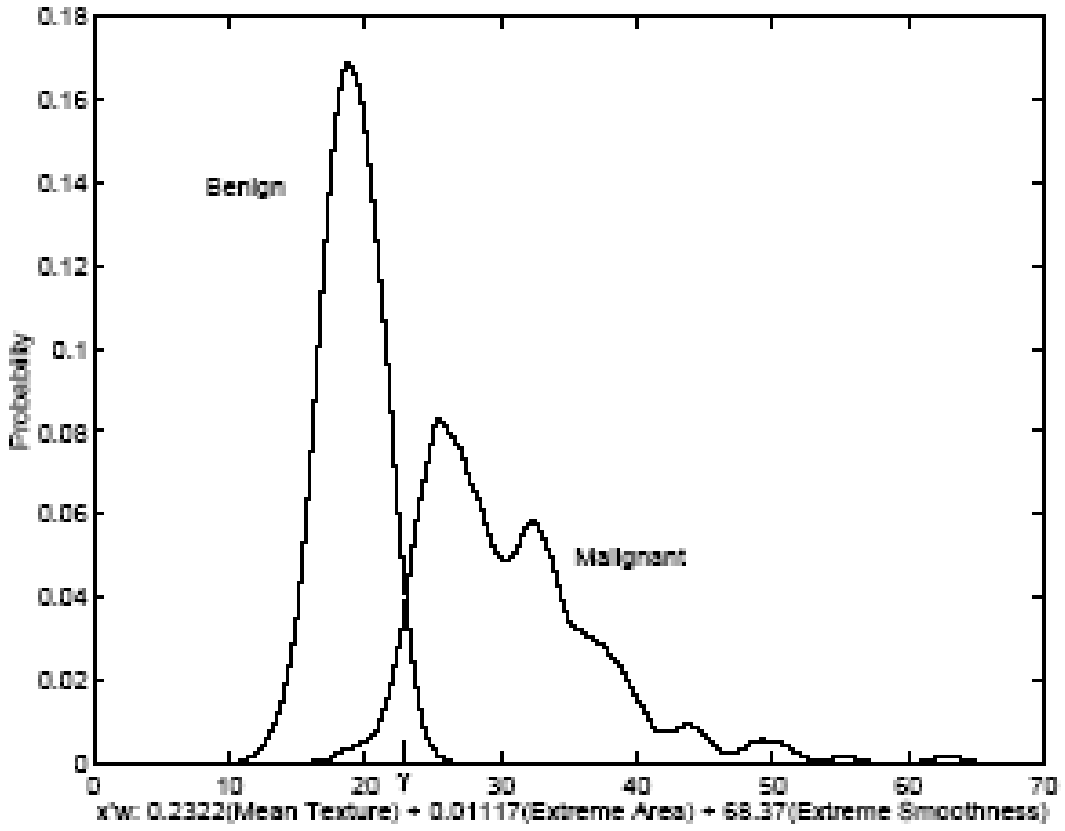


Figure 4-6. Densities of benign and malignant points along the normal ω to the separating plane $x^T\omega = \gamma$.

At the time of this paper (1994), there were 569 patients for the training set and the testing was on 131 subsequent patients in the database rather than the 699 that we used. The Xcyt program generates a 30-dimensional vector for each instance: (10 original features per image) x (3 other values computed from each original feature).

The program finds a separating plane and, if there are errors, can be recursively applied to each of the half-spaces previously found. The separating plane is shown here in Figure 4-7 (originally labelled Figure 2) reproduced from [10]. P1 is the original separating plane, with P2 and P3 found by recursion. No details of this figure explain why it is 2-dimensional, so we conclude it is to illustrate the separating plane recursion.

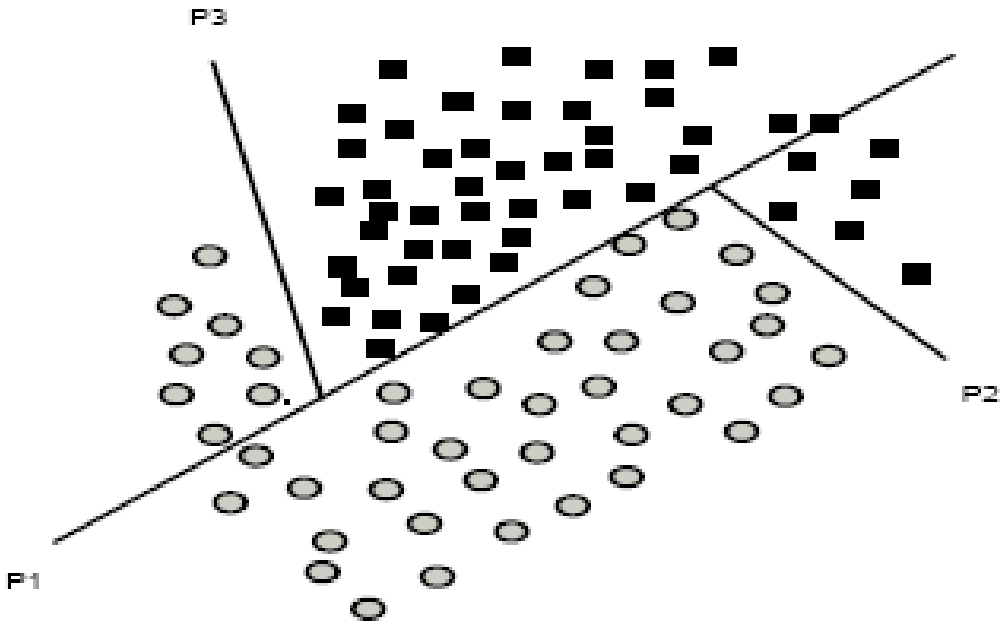


Figure 4-7. MSM-T separating planes.

It is quite possible that the Hyperplane Algorithm could be used recursively on points in the overlap region, but that is beyond the scope of this study.

4.1.5 CONCLUSIONS

According to our classification procedure, we expect the Hyperplane Algorithm to give the highest classification accuracy, which it does for this dataset. As this was part of our original two real-life datasets, *Hypothesis 1* was evaluated using *highest accuracy* for the

local version to determine if the local version could out-perform the global version. *Hypothesis 1* proved true for the local version. *Hypothesis 2* is verified only for the *hypercube*. *Hypothesis 3* proved true.

4.2 PIMA INDIANS DIABETES

The Pima Indian diabetes dataset is also a benchmark dataset available from the Information and Computer Science Department at the University of California, Irvine [21]. The Pima Indians dataset is usually a difficult dataset to classify due at least in part to the *noise* it contains. The Pima Indian dataset has 768 instances of eight attributes (none missing), plus a class attribute (diabetic or non-diabetic). Of the 768 instances, 268 are positive (diabetic) and 500 are negative (non-diabetic). Thus, about one third are positive and two thirds are negative. The class distribution and a short statistical analysis are shown in Tables 4-9 and 4-10, respectively.

All patients were Pima Indian females at least 21 years old. The nine attributes are:

1. Number of times pregnant
2. Plasma glucose concentration at 2 hours in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
9. Class: 1 for *tested positive for diabetes*, 2 otherwise

Missing Attribute Values: None

Table 4-9. Pima Indians Diabetes: Class Distribution

Class Distribution – Pima Indians Diabetes	
Class Value	**Number of instances**
0	500 (65.1%)
1	268 (34.9%)

Table 4-10. Pima Indians Diabetes: Statistical Analysis.

Brief statistical analysis				
Attribute number	Minimum	Maximum	Mean	Standard Deviation
1	0	17	3.8	3.4
2	0	199	120.9	32.0
3	0	122	69.1	19.4
4	0	99	20.5	16.0
5	0	846	79.8	115.2
6	0	67.1	32.0	7.9
7	0.078	2.42	0.5	0.3
8	21	81	33.2	11.8

4.2.1 HYPERPLANE ALGORITHM

The results (shown in Table 4-11 and Figure 4-8) are 64.94% average accuracy of classification when errors during training are not allowed. The hyperplane has simply been placed where all data on one side of it is classified as Class *1*, the majority class. By allowing 20% of Class *1* and 42% of Class *2* to be incorrectly classified during training, the average accuracy on a test set improves by ~4.5%; in this case, accuracy of Class *1* goes down and accuracy of Class *2* goes up.

Table 4 -11. Pima Indians Diabetes (5-fold cross-validation): Hyperplane Algorithm.

PPCP Algorithm	Pima Indians Diabetes Hyperplane Algorithm			
Error Allowed During Training:	0%,0%	20%,20%	40%,40%	20%,42%
Total: % Correct	64.94	64.94	66.23	69.48
Class 1: % Correct	100.00	100.00	71.00	78.00
Class 2: % Correct	0.00	0.00	57.41	53.70

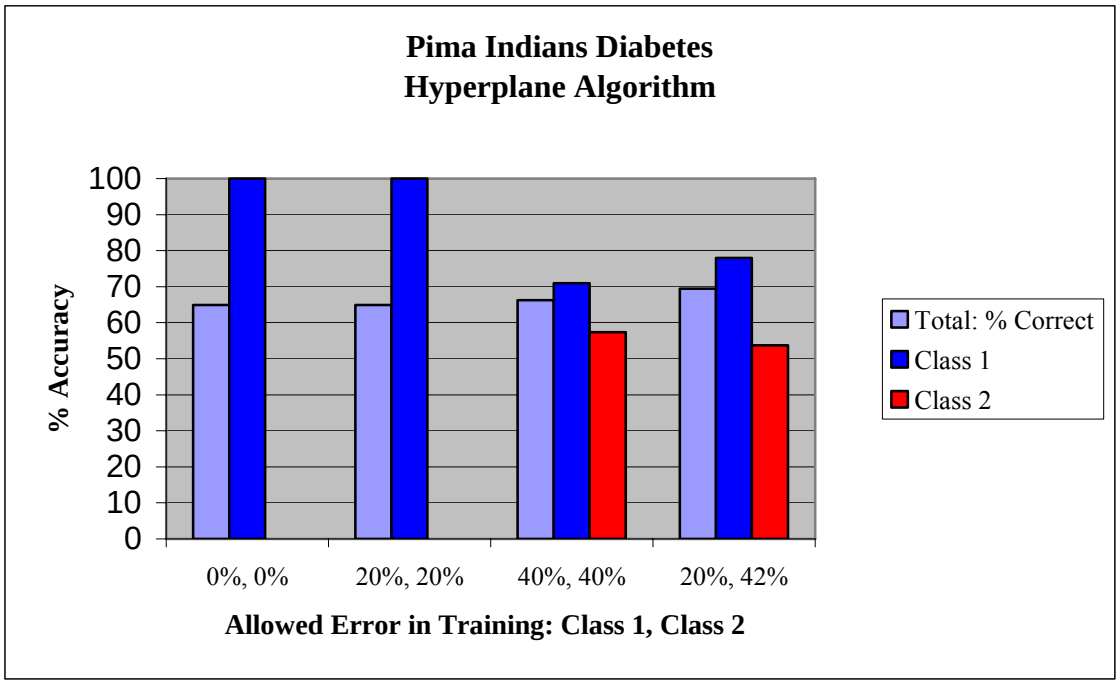


Figure 4-8. Pima Indians Diabetes (5-fold cross-validation): Hyperplane Algorithm.

It appears that as we approach the value of the estimated overlap (without outliers) as our error allowed during training, we also approach our maximum classification accuracy by this algorithm. This is also true for the Wisconsin Breast Cancer dataset.

According to our classification procedure, we would now go to the step 2 (the Margin Algorithm). Following this, we would decide whether the Hyperplane Algorithm or the Margin Algorithm would the classifier of choice. Though the Box Algorithm would not be used normally, we do so in order to evaluate the classification procedure.

4.2.2 MARGIN ALGORITHM

We present results for each, then compare the two versions.

Local Version

Figure 4-9 shows the progress of the algorithm as it steps through the two *for loops* [24]. A local maximum classified correctly is found for each inner loop. The maximum of these local maxima is the *best* local maximum correctly classified. The algorithm uses the classification parameters for this *best* local maximum.

Table 4-12 shows the learning constants found for each attribute during a typical training run [24]. Rather than the same value of the learning constant for all attributes of a class, individual values are found based on the classification performance of just that attribute. These values were the learning constants calculated at the maximum of the local maxima shown in Figure 4-9.

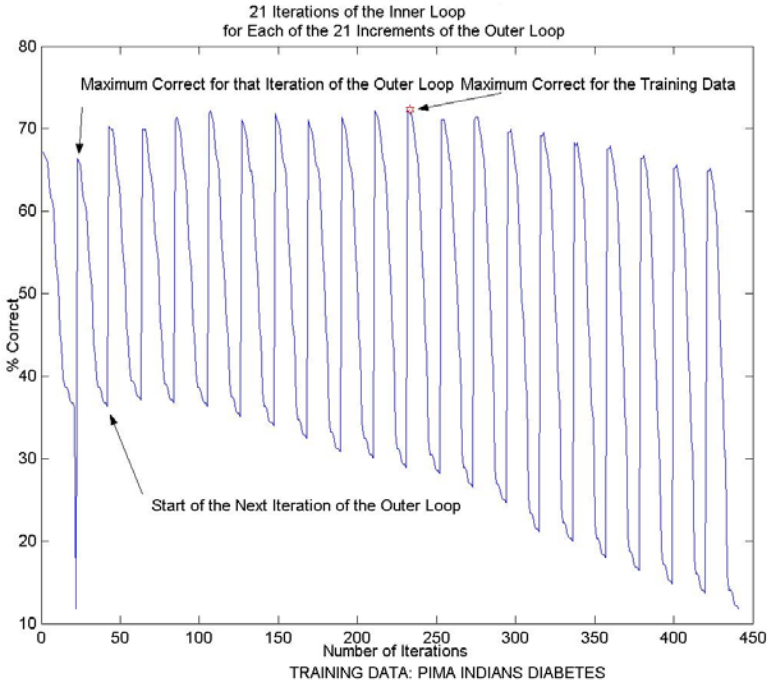


Figure 4-9. 2-D graph of the progression of the Margin Algorithm (local version) [24].

Table 4-12. Pima Indians Diabetes – learning constants and % correctly classified by attribute *i* individually [24].

Attribute #*i*	η^{i}_{A}	η^{i}_{B}	% Correct of Test Data by Attribute *i*
1	0.4	0	62.5
2	0.8	0.3	75.9
3	0	0	49.0
4	0.2	0	56.1
5	0.4	0.1	67.2
6	0.7	0	64.4
7	0.5	0	63.2
8	0.3	0.2	65.2

In Table 4-13, results for ten runs of training data for Pima Indians Diabetes dataset is shown [24]. Approximately one third of the entire dataset was used for training data, on each run, two thirds as testing data. The rank order of classification accuracy (most-to-least) by attributes was used.

Table 4-13. Pima Indians Diabetes:

10 runs of the Margin Algorithm (local version) [24].

Run #	% Correct of Test Data
	Local Version
1	71.4
2	73.7
3	74.7
4	75.8
5	74.2
6	75.3
7	73.4
8	75.3
9	75.3
10	73.7
Mean:	74.3

Figure 4-10 shows the classification accuracy on testing, using the local version of the Margin Algorithm, and averaging ten runs when the size of the training set varies from 10% to 70%of the entire dataset.

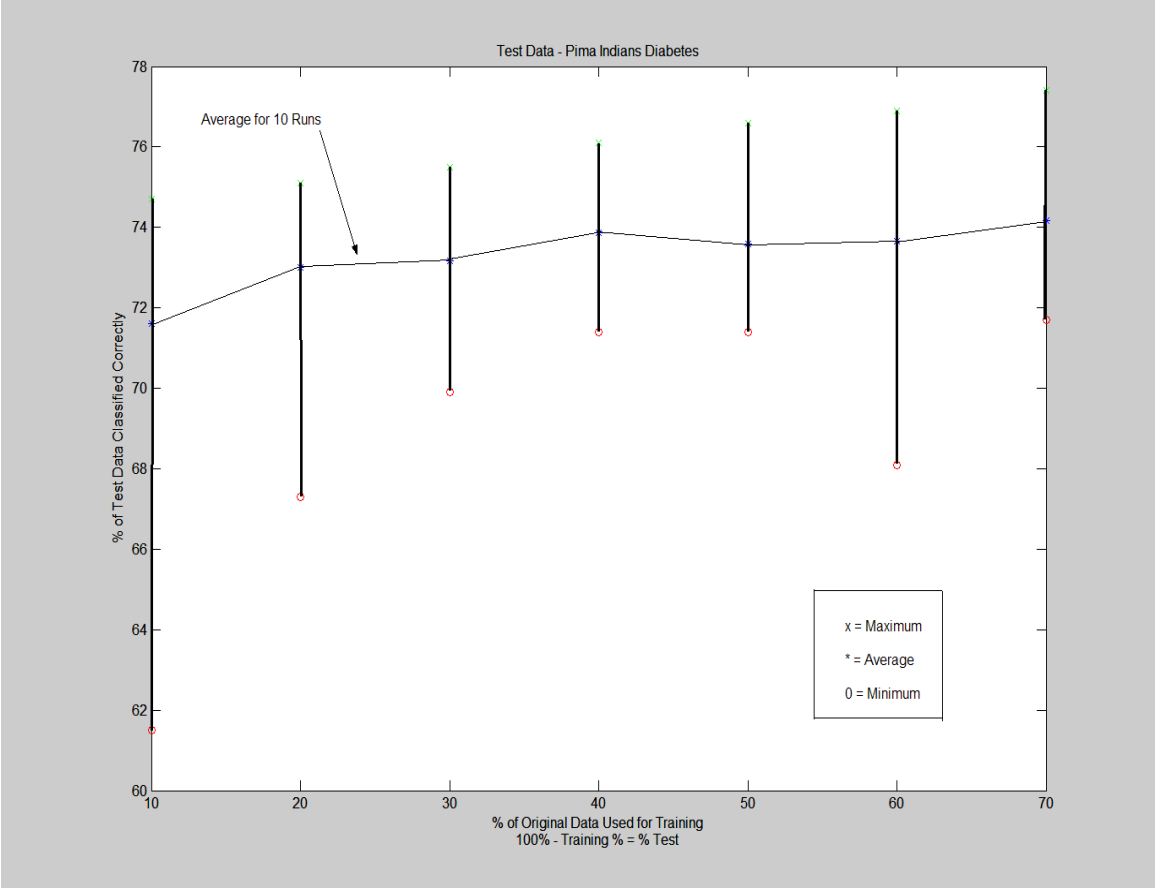


Figure 4-10. Pima Indians Diabetes: average accuracy vs. % of training data

Margin Algorithm (local version).

Global Version

Table 4-14 shows the ranges of values for both classes and the extensive overlap between classes [25]. The overlap makes separation appear difficult.

Table 4-15 shows the ranges of values for both classes after overlap are limited by the margins created by Margin (global version) [25]. The overlap between classes is no longer extensive. Note that the minimum values for Class *1* is unchanged, as are the maximum values for Class *2*. The margins are the regions of overlap for each attribute.

Table 4-14. Pima Indians Diabetes: ranges for attribute values for each class [25].

	Class 1 (Diabetic)		Class 2 (Non-diabetic)	
Attribute #	Minimum Value	Maximum Value	Minimum Value	Maximum Value
1	0	13	0	17
2	0	197	0	199
3	0	122	0	114
4	0	60	0	99
5	0	744	0	846
6	0	57.3	0	67.1
7	0.078	2.329	0.088	2.42
8	21	81	21	70

Table 4-15. Pima Indians Diabetes: truncated ranges for attribute values for each class after training by the Margin Algorithm (global version) to create the margins [25].

	Class 1 (Diabetic)		Class 2 (Non-diabetic)	
Attribute #	Minimum Value	Maximum Value	Minimum Value	Maximum Value
1	0	7.4039	2.9009	17
2	0	157.71	107.83	199
3	0	80.705	63.187	114
4	0	29.951	17.892	99
5	0	166.19	62.108	846
6	0	37.025	28.602	67.1
7	0.078	0.72941	0.41378	2.42
8	21	43.735	29.172	70

In this dataset, Class *1* is to the left of Class *2*, relative to the origin. Separation of classes proceeds by classifying the points outside the margin as belonging to Class *1* or Class *2*. The overlaps between the ranges shown in Table 4-14 are the margins and explicitly shown in Table 4-16.

Table 4-16. Margins for Pima Indians Diabetes.

Attribute #	Left margin	Right margin
1	2.9009	7.4039
2	107.83	157.71
3	63.187	80.705
4	17.892	29.951
5	62.108	166.19
6	28.602	37.025
7	0.41378	0.72941
8	29.172	43.735

A surface plot of the classification accuracy on training data is shown in Figure 4-11. The percentage correctly classified is plotted versus $\eta_A\sigma_1$ and $\eta_A\sigma_2$. For this graph, one third of the dataset is being used for training and the balance for testing.

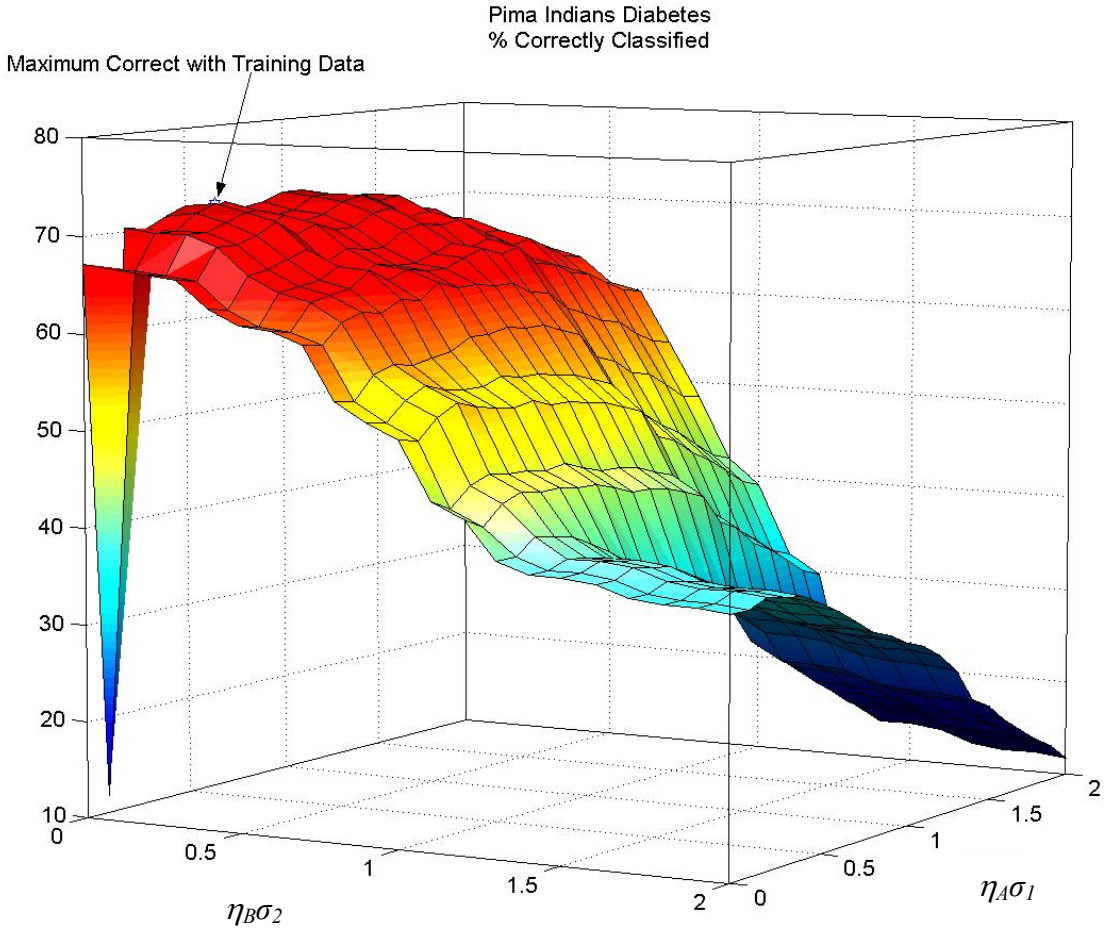


Figure 4-11. Pima Indians Diabetes: Margin Algorithm (global version)

3-D plot of percentage correctly classified vs. $\eta_A\sigma_1$ and $\eta_B\sigma_2$.

Comparison of Versions

Using the same training data, the learning constants were found for each version, then used on the same test set. Results over ten runs are shown in Table 4-17.

On a trial run, the local version classifies the test data with 73.6% accuracy, which compares favorably with other results in the literature. All points were classified, i.e., none were left in the margins. The percentage classified correctly is slightly less than that of the best single classifier as calculated for that run. On another run, the test results

are 75.7%, which is somewhat better than the single best classifier results of 71.1% as calculated for that run.

Table 4-17. Pima Indians Diabetes – 10 runs comparing the two versions[25].

Run #	% Correct of Test Data	% Correct of Test Data
	Local Version	Global Version
1	71.4	82.5
2	73.7	77.5
3	74.7	83.1
4	75.8	75.7
5	74.2	81.6
6	75.3	81.9
7	73.4	77.5
8	75.3	75.9
9	75.3	79.8
10	73.7	77.5
Mean:	74.3	79.3

4.2.3 BOX ALGORITHM

In initial testing, the results range (shown in Figure 4-12) from 71% to 75% average accuracy of classification for the Pima Indians Diabetes dataset for the initial testing (*cube* version) with 100 trials of 50:50 splits of the data between training and testing [26].

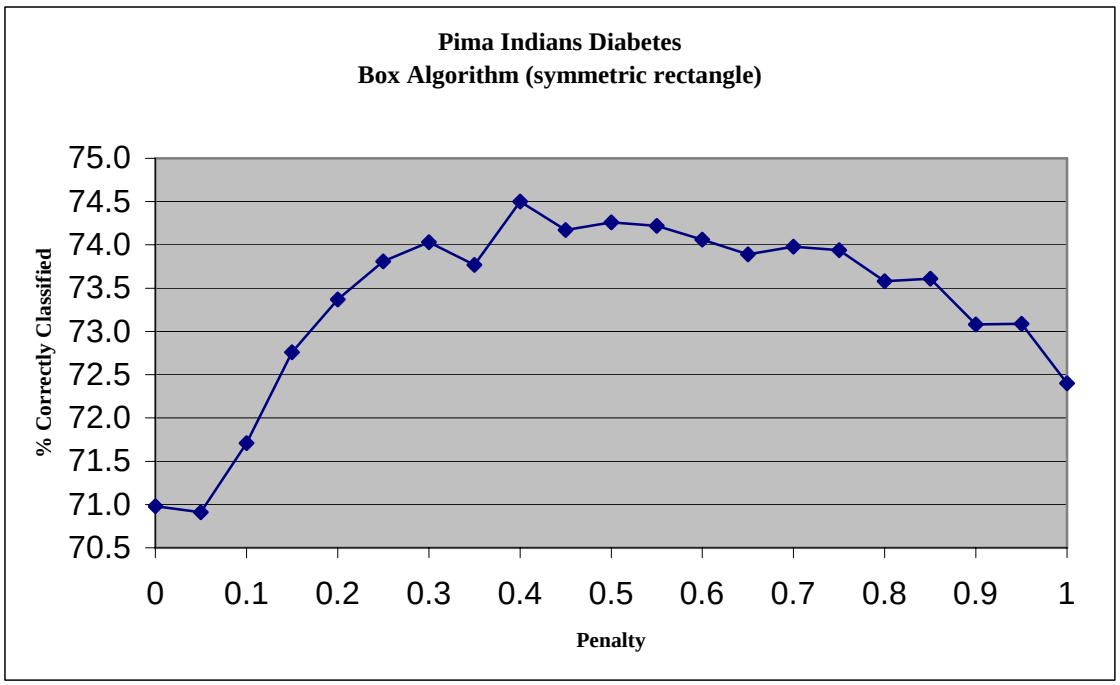


Figure 4-12. Results averaged over 100 trials for each value of the penalty tested [26].

In this initial testing, a *penalty*, for misclassification of points, $0 \leq$ *penalty* ≤ 1 by a step size of 0.1 was evaluated. Figure 4-12 shows the results. At *penalty* = 0.25, the classification accuracy is near the maximum. At *penalty* > 0.6, the classification accuracy appeared to decrease. The highest classification accuracy appears to be when $0.3 <$ *penalty* < 0.75. Based on similar results for the Wisconsin Breast Cancer dataset, a *penalty* = 0.4 was deemed reasonable to use on all other sets to be tested in this study.

Subsequently, with 5-fold cross-validation, the *hypercube* version had an average total accuracy of ~65% (shown in Table 4-18 and Figure 4-13) with each of the two orders of classes used to classify. In both orders, Class *1* was ~88% accurate and Class *2* was ~24% accurate.

Table 4-18. Pima Indians Diabetes (5-fold cross-validation):

Box Algorithm (cube).

PPCP Algorithm	Pima Indians Diabetes Box Algorithm (cube)	
Class Order	1,2	2,1
Total: % Correct	65.40	65.10
Class 1: % Correct	88.00	87.13
Class 2: % Correct	23.56	24.31

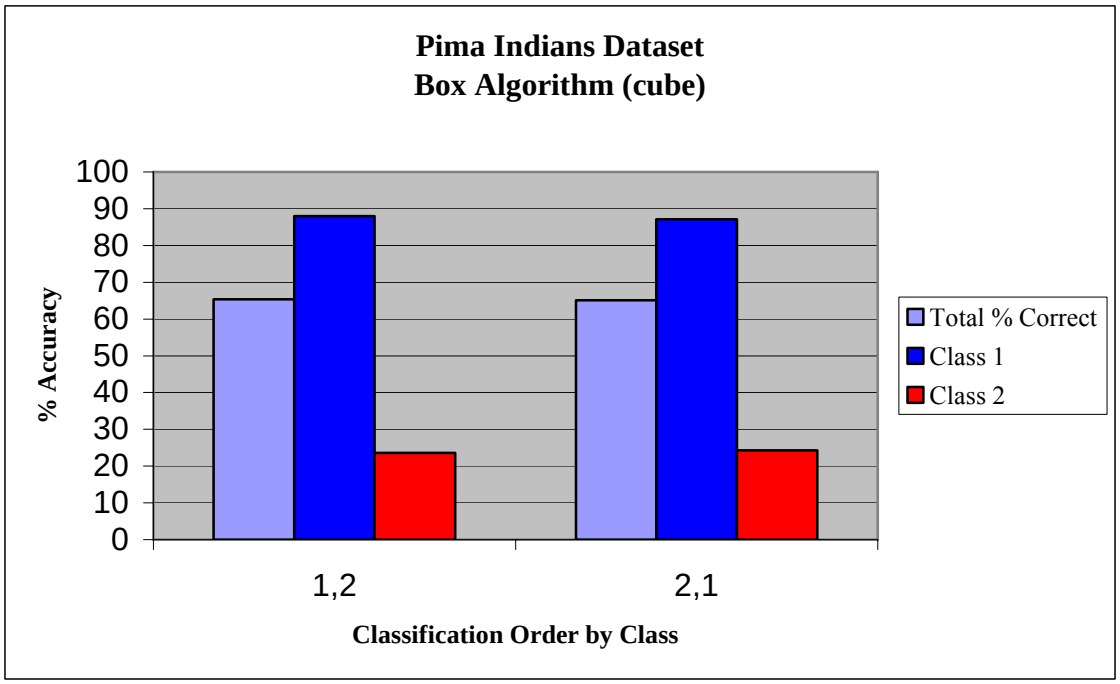


Figure 4-13. Pima Indians Diabetes (5-fold cross-validation):

Box Algorithm (cube).

Table 4-19 and Figure 4-14 show the results for the *symmetric rectangle*. When the order was Class *1* followed by Class *2*, the *symmetric rectangle* had an average total accuracy of ~68%. When Class *1* was followed by Class *2*, Class *1* was ~90% accurate and Class *2* was ~28% accurate.

Table 4-19. Pima Indians Diabetes (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

PPCP Algorithm	Pima Indians Diabetes Box Algorithm (symmetric rectangle)	
Class Order	1,2	2,1
Total: % Correct	68.05	33.64
Class 1: % Correct	89.60	0.00
Class 2: % Correct	28.15	95.93

All of the accuracies mentioned to this point showed an increase in accuracy of 1-5% over the *hypercube* version. However, in the order of Class *2* followed by Class *1*, Class *1* was ~90% accurate and Class *2* was ~28% accurate. The average total accuracy dropped to ~33% while the accuracy of Class *2* was greatly increased.

It appears that if Class *1* is modeled well, Class *2* is not, and vice versa. This indicates a large overlap, as estimated by the Hyperplane Algorithm.

The extent of the box for Class *2* was allowed to go to 5 standard deviations rather than the usual 3. This was done because with 3 standard deviations, less than 5% of Class *2* was accurately classified. An assumption of a normal distribution here is not justified. The *symmetric rectangle* more successfully models Class *2* than the *hypercube* does when using the class order 1→2 while slightly improving the classification accuracy of Class *1*.

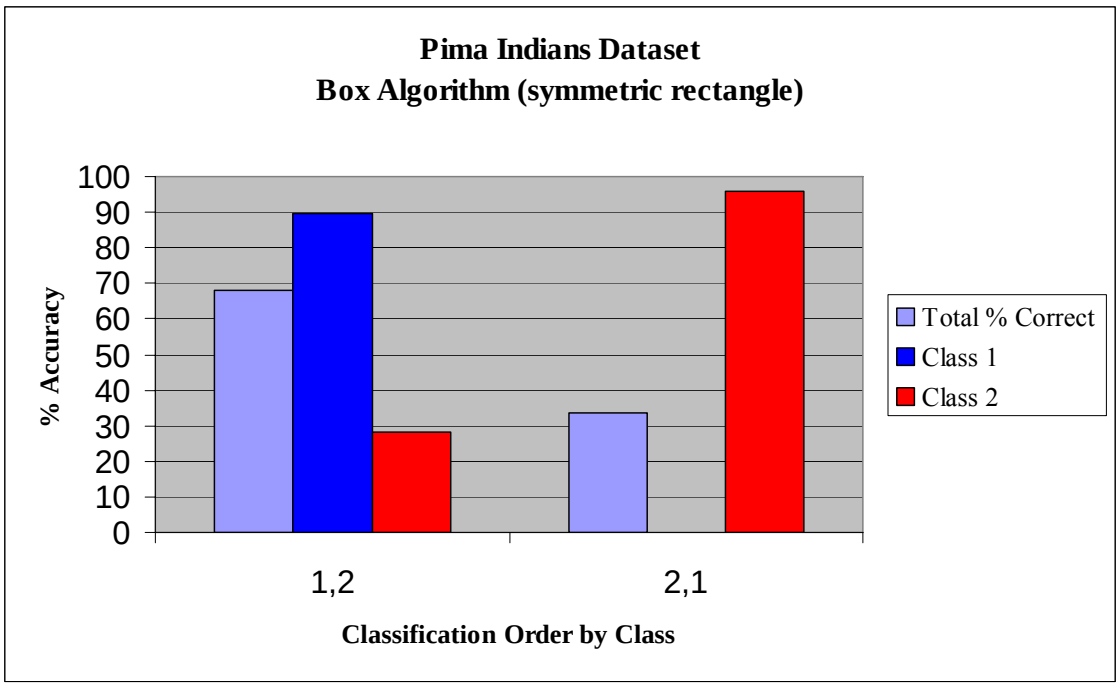


Figure 4-14. Pima Indians Diabetes (5-fold cross-validation): Box Algorithm (symmetric rectangle).

All of the accuracies mentioned to this point showed an increase in accuracy of 1-5% over the *hypercube* version. However, in the order of Class 2 followed by Class *1*, Class *1* was ~90% accurate and Class *2* was ~28% accurate. The average total accuracy dropped to ~33% while the accuracy of Class *2* was greatly increased. It would seem that the box that approximates Class *2* completely encloses Class *1*. Even so, it is still not large enough to include all points of Class *2*. This is quite remarkable in view of the fact that the extent of the box for Class *2* was allowed to go to 5 standard deviations rather than the usual 3. This was done because with 3 standard deviations, less than 5% of Class *2* was accurately classified. An assumption of a normal distribution here is not justified. The *symmetric rectangle* more successfully models Class *2* than the *hypercube* does.

When there is heavy overlap, *Hypothesis 2* (class order for classification) holds true for the *symmetric rectangle*.

Overall classification accuracy is increased for class order 1→2 when the *symmetric rectangle* is used. As the better classification order by class would be used, *Hypothesis 3* (symmetric rectangles classify better than hypercubes) holds then.

4.2.4 DISTRIBUTIONS OF THE CLASSES

We again look at a graph of the estimated distributions of the two classes (shown in Figure 4-15). As in section *4.1.4*, by means of a histogram, an estimate of these distributions was created using the trial data. This time we do not normalize the vector connecting the means of the two classes, but simply show the estimated distributions relative to one another.

Unlike the case of the Wisconsin Breast Cancer, the distributions for the two classes overlap greatly, with the majority of examples for *both* classes in the region of overlap. It is not possible to choose a value to separate the two classes with high classification accuracy. This is why the Hyperplane Algorithm is a poor classifier for this dataset. The Box Algorithm can also do better when given such distributions for the two classes. If one class were totally inside another, the distribution of one class would be totally inside the distribution for the other class. This is not true here, so we think this explains why the Margin Algorithm step more accurately classifies this dataset than other steps.

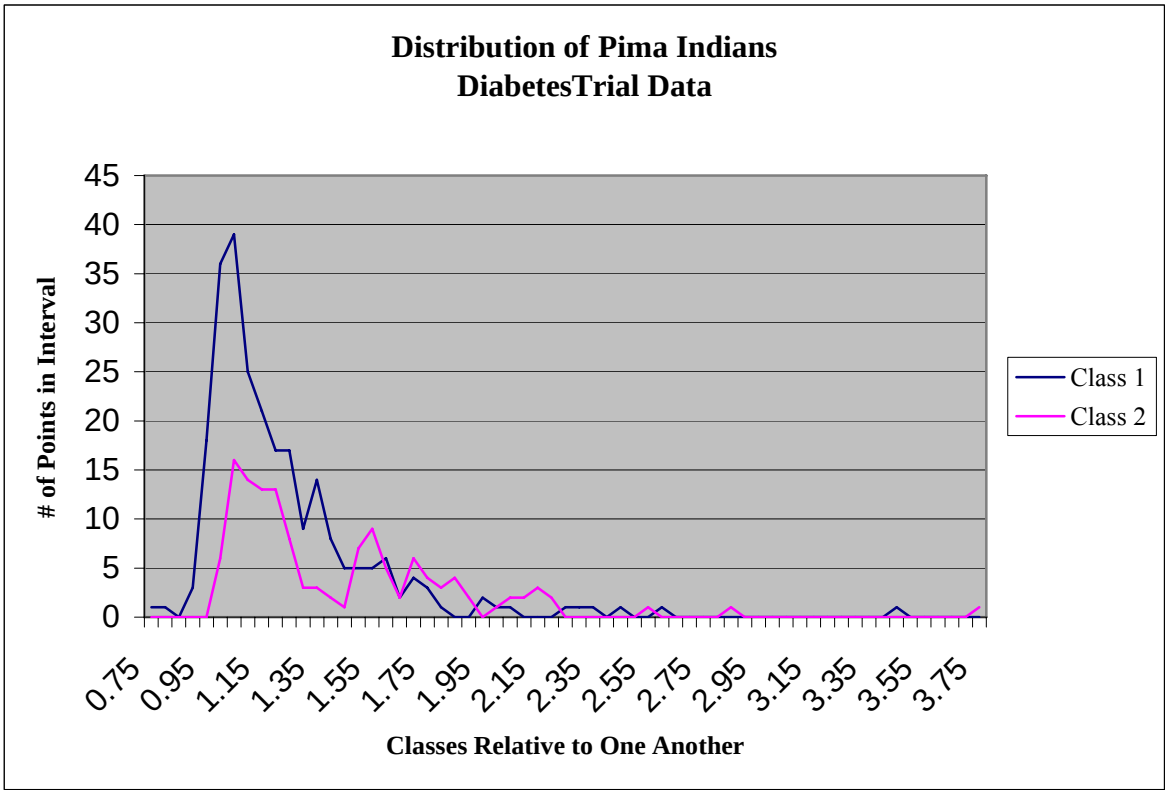


Figure 4-15. Pima Indians Diabetes:

Approximate Distribution Curves of the Two Classes.

4.2.5 CONCLUSIONS

According to our classification procedure, the Margin Algorithm would give the highest classification accuracy, which it does for this dataset. As this was part of our original two real-life datasets, *Hypothesis 1* was evaluated using *highest accuracy* for the local version to determine if the local version could out-perform the global version. *Hypothesis 1* proved true for the local version. *Hypothesis 2* is verified only for the *symmetric rectangle*. *Hypothesis 3* proved true only when the class order was 1→2.

4.3 IRIS

The Iris diabetes dataset is a benchmark dataset available from the Information and Computer Science Department at the University of California, Irvine [21]. The data is composed of four attributes, plus a class attribute. The Iris dataset contains three classes of fifty instances each, where each class refers to a type of iris plant (Iris Setosa, Iris Versicolor, Iris Virginica). Thus, the classes are evenly distributed. One class is linearly separable from the other two; the latter are *not* linearly separable from each other. The class distribution and a short statistical analysis are shown in Tables 4-20 and 4-21, respectively.

This is perhaps the best-known database to be found in the pattern recognition literature. Fisher's paper is a classic in the field and is referenced frequently to this day. (See Duda & Hart, for example.) This data differs from the data presented in Fisher's article (identified by Steve Chadwick, spchadwick@espeedaz.net): *"The 35th sample should be: 4.9,3.1,1.5,0.2,"Iris-setosa" where the error is in the fourth feature. The 38th sample: 4.9,3.6,1.4,0.1,"Iris-setosa" where the errors are in the second and third features."*

The five attributes are:

1. sepal length in cm
2. sepal width in cm
3. petal length in cm
4. petal width in cm
5. class: Iris Setosa, Iris Versicolor, Iris Virginica

Missing Attribute Values: None

Table 4-20. Iris: Class Distribution.

Class Distribution - Iris	
Class Value	**Number of instances**
Iris Setosa	50 (33.3%)
Iris Versicolor	50 (33.3%)
Iris Virginica	50 (33.3%)

Table 4-21. Iris: Statistical Analysis.

Brief statistical analysis					
Attribute	Minimum	Maximum	Mean	Standard Deviation	Correlation
sepal length	4.3	7.9	5.84	0.83	0.7826
sepal width	2.0	4.4	3.05	0.43	-0.4194
petal length	1.0	6.9	3.76	1.76	0.9490
petal width	0.1	2.5	1.20	0.76	0.9565

4.3.1 HYPERPLANE ALGORITHM

The classification accuracy (shown in Table 4-22 and Figure 4-16) ranges from 68.40% when no error is allowed during training to 90.50% when 0%, 15%, and 20% error is allowed for Class *1*, Class *2*, and Class *3*, respectively, during training. Figure 4-16 shows results with differing amounts of error allowed during training.

Table 4-22. Iris (5-fold cross-validation): Hyperplane Algorithm.

PPCP Algortihm	Iris - Hyperplane Algorithm					
Error Allowed During Training: Class 1, Class 2, Class 3	0%, 0%, 0%	0%, 0%, 5%	0%, 0%, 15%	0%, 0%, 25%	0%, 15%, 15%	0%, 15%, 20%
Total: % Correct	68.40	84.00	85.97	87.33	90.20	90.50
Class 1: % Correct	97.70	97.10	97.50	100.00	98.20	98.10
Class 2: % Correct	13.50	82.70	84.90	90.00	86.00	87.10
Class 3: % Correct	94.00	72.20	75.50	72.00	86.40	86.30

When no error is allowed during training, Class *1* (the separable class) and Class *3* are classified with high accuracy. As we increase the allowed error during training, the classification accuracy of Class *3* decreases while the classification accuracy of Class *2* increases. These two classes are not linearly separable. By allowing error during training for these two classes, overall classification accuracy is 90%+, with classification accuracy of 98%+ for Class *1* and classification accuracy of 86%+ for Class *2* as well as Class *3*.

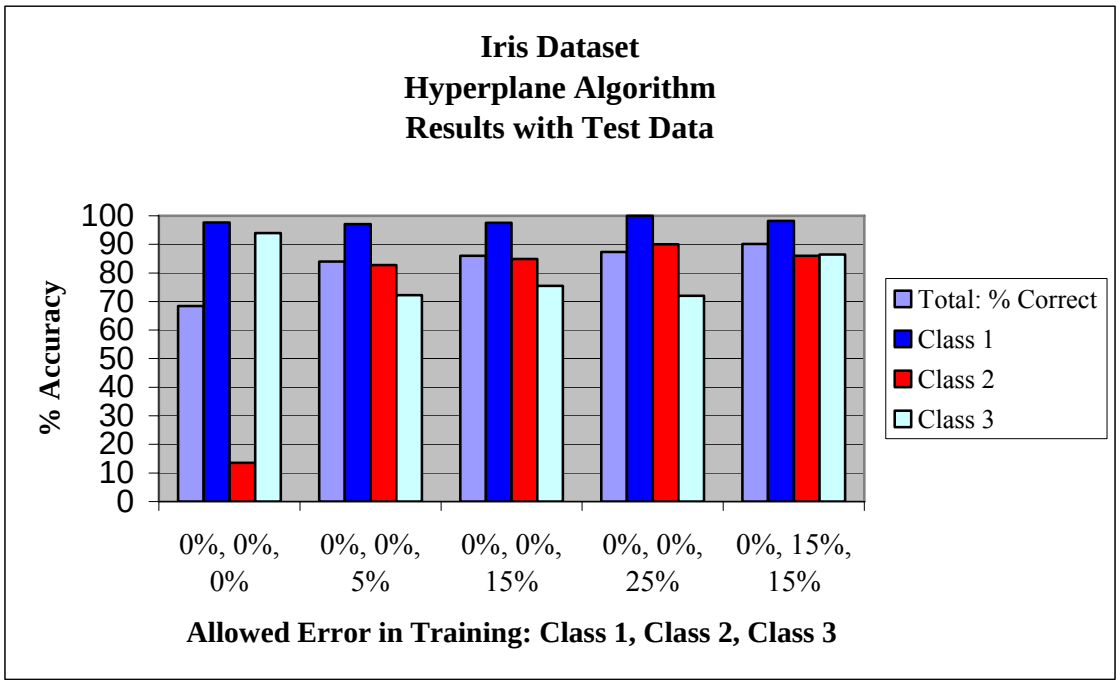


Figure 4-16. Iris (5-fold cross-validation): Hyperplane Algorithm.

According to our classification procedure, we would now go to the step 2 (the Margin Algorithm). Following this, we would decide whether the Hyperplane Algorithm or the Margin Algorithm would be the classifier of choice. Though the Box Algorithm would not be used normally, we do so in order to evaluate the classification procedure.

4.3.2 MARGIN ALGORITHM

Previous work [25] showed that Margin is feasible for two-class classification where for each attribute the mean for Class *1* is to the left of the mean for Class *2*. Both a global version [25] and a local version [26] were tested.

For Iris, Margin is extended beyond two-class classification and the means may be right-to-left or left-to-right. Figure 4-17 shows the process for a simplified version.

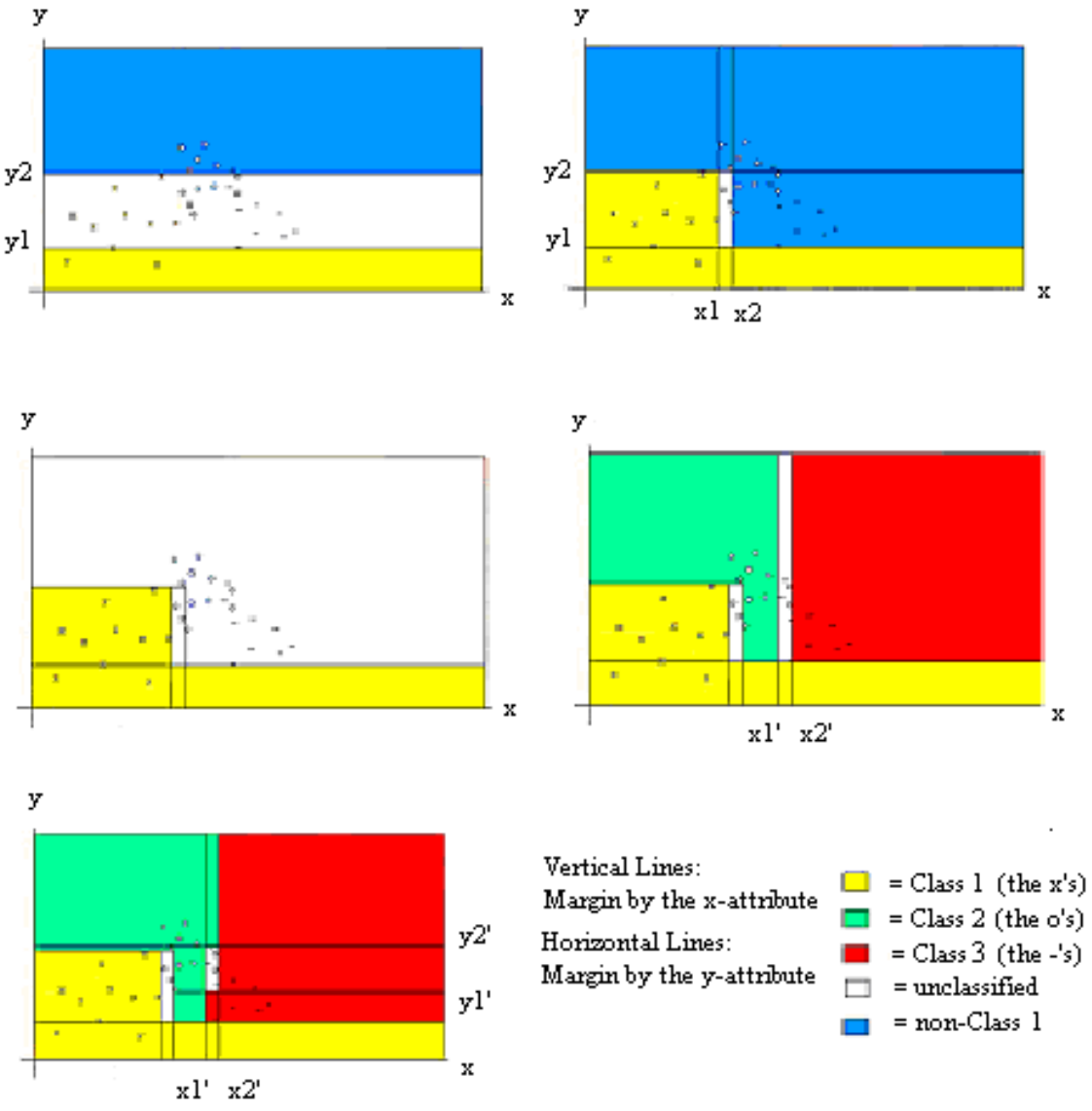


Figure 4-17. Illustration of Margin's Process:

Class *1* vs. non-Class *1* → Class *2* vs. Class *3*.

***Example* 3:** Consider the artificial dataset with three classes shown in Figure 4-17. This is a simple version of the algorithm and the margin edges are not in the final positions. Starting with the *y*-attribute, the margin edges (y_1, y_2) are found and the classification is made for Class *1* versus non-Class *1*. Then the margin edges (x_1, x_2) are found for the x-attribute and further classification is made for Class *1* versus non- Class *1*. In *top ,right* of Figure 4-17, the lower L-shaped area is Class *1* and the upper L-shaped area is non-Class *1*. The small rectangle is *unclassified.* The margins (x_1', x_2') and (y_1', y_2') are used to further classify points in non-Class *1* as Class *2* versus Class *3*. *Bottom, left* of Figure 4-17 shows the result. Note that there is a second small rectangle that is *unclassified*.

Two versions of the algorithm are tested: a global version and a local version. For the local version used, there is a best order of attributes using rank order of classification, i.e., attributes are used in decreasing order of their *highest accuracy*. The rank order classification ability (from highest-to-lowest) is 4→3→1→2, as was determined during training. The small number of attributes to test how the order of attribute use in classification affects the accuracy. We present results (shown in Table 4-23) for each, then compare the two versions.

Local Version

With 100 trial runs for each of the 24 permutations, the results range from 63.32% to 95.57% average accuracy of classification. There is small variation within most groupings and the overall variation is large.

Global Version

With 100 trial runs for each of the 24 permutations, the average classification accuracy ranges from 88.89% to 94.33%. There is little variation within each of the groupings

shown starting with a particular attribute and the overall variation is much less than with the local version.

Table 4-23. Iris: Margin Algorithm (global version vs. local version).

Permutations of the 4 attributes – order of use for classification	Average % Classified Correctly (100 trials each permutation)	
	Global Version	Local Version
	Ave Correct % (std deviation %)	Ave Correct % (std deviation %)
1,2,3,4	90.79 (4.95)	71.24 (3.46)
1,2,4,3	92.64 (4.18)	71.15 (3.23)
1,3,2,4	92.47 (3.84)	71.45 (3.57)
1,3,4,2	**93.64 (4.13)**	71.37 (3.21)
1,4,2,3	92.83 (3.54)	**71.77 (3.32)**
1,4,3,2	92.04 (5.20)	71.24 (3.45)
2,1,3,4	88.89 (5.13)	64.48 (5.59)
2,1,4,3	89.17 (4.54)	63.32 (6.12)
2,3,1,4	89.47 (5.07)	**69.91 (6.76)**
2,3,4,1	**91.60 (4.43)**	67.35 (6.88)
2,4,1,3	89.92 (4.36)	69.28 (8.08)
2,4,3,1	89.84 (5.00)	69.52 (7.86)
3,1,2,4	92.77 (3.77)	90.83 (3.83)
3,1,4,2	93.68 (3.08)	90.97 (4.30)
3,2,1,4	93.44 (4.50)	92.59 (3.18)
3,2,4,1	94.09 (3.72)	92.51 (3.39)
3,4,1,2	**94.43 (3.88)**	93.60 (3.23)
3,4,2,1	94.11 (3.75)	**93.99 (2.74)**
4,1,2,3	93.77 (3.88)	93.99 (3.11)
4,1,3,2	94.08 (3.99)	94.08 (2.94)
4,2,1,3	**94.41 (3.45)**	94.45 (2.46)
4,2,3,1	94.19 (3.36)	95.44 (1.77)
4,3,1,2	***94.01 (3.79)***	***94.69 (2.98)***
4,3,2,1	94.33 (2.97)	**95.57 (1.81)**

Comparison of Versions

The maximum classification accuracy using each of the four attribute as the initial classifier is highlighted in bold print in Table 4-23. The attribute order that was predicted by *highest accuracy* (4, 3, 1, 2) is highlighted in italicized print. As hypothesized, the order of attribute used during classification is important, particularly for the local version of the Margin Algorithm.

Clearly, attribute order is of great importance in the local version, but it does not appear to be totally explained by rank order by single components. If it did, we might reasonably expect 4, 3, 1, 2 (4→3→1→2) to be better than all other permutations. This is shown false by counterexamples: 4, 2, 3, 1 gives a better (average) classification accuracy, as does 4, 3, 2, 1. However, the difference is less than 1% and may simply be a statistical anomaly.

The two versions were quite close for the overall classification accuracy with the predicted best order, i.e., by *highest accuracy* as well as with each version's true best order. If the processing time for sorting the attributes is a concern, as with very large *k*, one may choose the global version.

According to our classification procedure, we would now stop and return classification parameters for the Margin Algorithm.

4.3.3 BOX ALGORITHM

A penalty of 0.4 was used in the training, as this appears from previous testing with two real datasets to be an optimal value. With 5-fold cross-validation, *Hypothesis 2* (class order) and *Hypothesis 3* (*hypercubes* versus *symmetric rectangles*) were tested and results are shown in Table 4-24 and Figure 4-18.

For the *cube*, as can be seen in the top row of Table 4-23, roughly 80% of the data is classified correctly by three of the orders and roughly 87% is classified correctly by the other three of the orders. The order of classification for the second and third classes affects the overall results. When Class *2* is classified before Class *3*, the total accuracy is better. Also apparent is that whichever of these two classes is classified first has better classification accuracy. Both of these observations lead to the conclusion that there is a substantial overlap between them.

Table 4-24. Iris (5-fold cross-validation): Box Algorithm (cube).

PPCP Algorithm	Iris - Box Algorithm (cube)					
Class Order	1,2,3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1
Total: % Correct	86.69	80.32	87.05	87.69	80.92	81.31
Class 1: % Correct	88.56	90.60	90.52	91.12	90.32	91.64
Class 2: % Correct	93.12	65.04	93.72	93.92	66.56	66.68
Class 3: % Correct	78.40	85.32	76.92	78.04	85.88	85.60

It is known that these two classes are non-separable. The Hyperplane Algorithm gave best classification accuracy when it was calculated allowing 15%+ errors during training. This is an inference of the degree of overlap. The observations regarding Class *2* and Class *3* tend to confirm this.

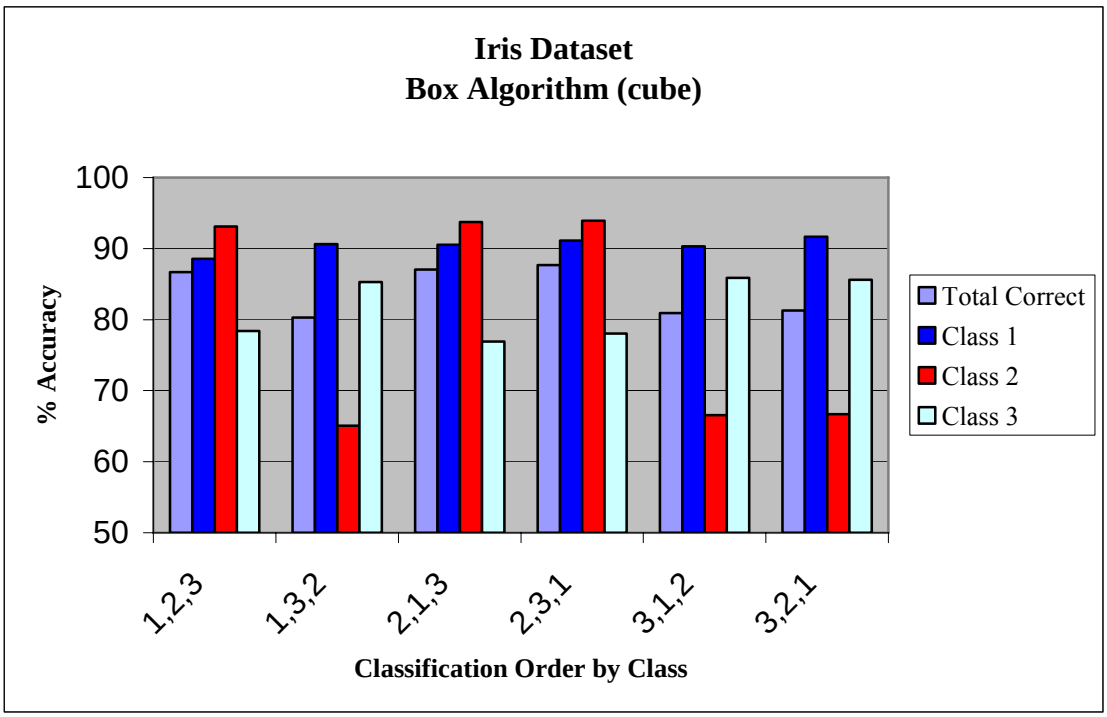


Figure 4-18. Iris (5-fold cross-validation): Box Algorithm (cube).

For the *symmetric rectangle*, the computational complexity increases quite a bit. Because of this, preliminary testing was used to roughly approximate the limits of the dimensions of the box. This was accomplished by using a larger step-size during the *for loops*. Results are shown in Table 4-25 and Figure 4-19.

The same patterns are generally seen in the data, but mitigated. Noticeably, the class order 3, 2, 1 has an improvement in accuracy of 7%. The usual improvement for *symmetric rectangle* over *cube* is 2-5%. This is similar to the results with the artificial sets in 4-dimensions.

Note that both the overall classification accuracy and the classification accuracy for a particular class are both affected by order of classification.

Table 4-25. Iris (5-fold cross-validation): Box Algorithm (symmetric rectangle).

PPCP Algorithm	Iris - Box Algorithm (symmetric rectangle)					
Class Order	1,2,3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1
Total: % Correct	88.53	84.44	89.07	90.40	85.87	88.67
Class 1:% Correct	90.80	89.33	88.00	87.60	87.20	94.00
Class 2: % Correct	92.80	77.33	94.80	95.20	80.00	84.00
Class 3: % Correct	82.00	86.67	84.40	88.40	90.40	88.00

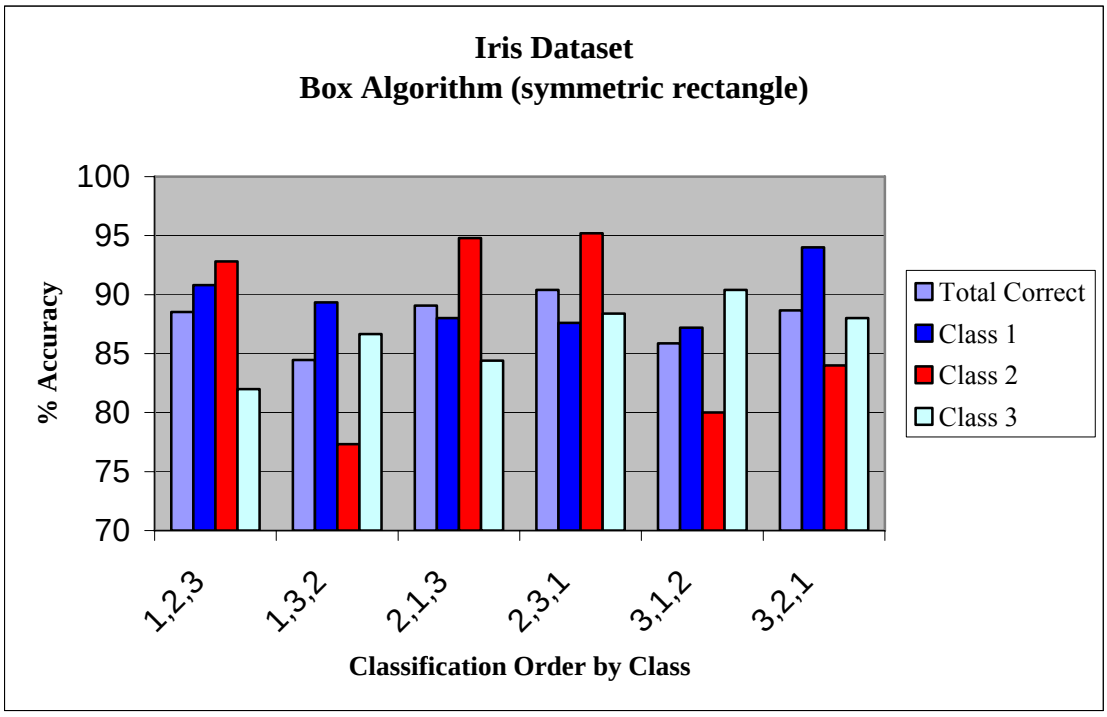


Figure 4-19. Iris (5-fold cross-validation): Box Algorithm (symmetric rectangle).

4.1.5 CONCLUSIONS

According to our classification procedure, we expect the Margin Algorithm to give the highest classification accuracy, which it does for this dataset. *Symmetric rectangles* classify more accurately than do *hypercubes*. *Hypothesis 1*, *Hypothesis 2*, and *Hypothesis 3* are verified.

4.4 STATLOG HEART DISEASE

The StatLog Heart Disease dataset is also a benchmark dataset available from the Information and Computer Science Department at the University of California, Irvine [21]. The StatLog Heart Disease has 270 instances of thirteen attributes (none missing) that have been extracted from a larger set of 75 attributes, plus a class attribute (disease or non-disease). Of the 270 instances, 120 are positive (disease) and 150 are negative (non-disease). Thus, about 44% are positive and 56% are negative. The class distribution and a short statistical analysis are shown in Tables 4-26 and 4-27, respectively.

The thirteen attributes are:

1. Age
2. Sex
3. Chest pain type (4 values)
4. Resting blood pressure
5. serum cholesterol in mg/dl
6. Fasting blood sugar > 120 mg/dl
7. Resting electrocardiographic results (values 0,1,2)
8. Maximum heart rate achieved
9. Exercise induced angina
10. Oldpeak = ST depression induced by exercise relative to rest
11. The slope of the peak exercise ST segment
12. Number of major vessels (0-3) colored by fluoroscopy
13. Thal: 3 = normal; 6 = fixed defect; 7 = reversible defect

Attributes types

Real: 1, 4, 5, 8, 10, 12

Ordered: 11

Binary: 2, 6, 9

Nominal: 3, 7, 13

Missing Attribute Values: None

Table 4-26. StatLog Heart Disease: Class Distribution.

Class Distribution - StatLog Heart Disease	
Class Value	**Number of instances**
0	150 (55.6%)
1	120 (44.4%)

Table 4-27. StatLog Heart Disease: Statistical Analysis.

Brief statistical analysis				
Attribute number	Minimum	Maximum	Mean	Standard Deviation
1	29	71	54.4	9.1
2	0	1	0.7	0.5
3	1	4	3.2	1.0
4	29	77	54.4	9.1
5	0	1	0.7	0.5
6	1	4	3.2	1.0
7	94	200	131.3	17.9
8	126	564	249.7	51.7
9	0	1	0.1	0.4
10	0	2	1.0	1.0
11	71	202	149.7	23.2
12	0	1	0.3	0.5
13	0	6.2	1.1	1.1

4.4.1 HYPERPLANE ALGORITHM

The classification accuracy (shown in Table 4-28 and Figure 4-20) ranges from 55.56% with no error allowed during training to 64.00% when 40% error for Class *1* and 30% error for Class *2* is allowed during training. Note that by simply choosing Class *1* every time to classify a generic point **x**, we can achieve 55.56% accuracy, i.e., this is the majority class.

According to our classification procedure, we would now go to step 2 (the Margin Algorithm). Following this, we would decide whether the Hyperplane Algorithm or the Margin Algorithm would be the classifier of choice. Though the Box Algorithm would not be used normally, once again we do so in order to evaluate the classification procedure.

Table 4-28. StatLog Heart Disease (5-fold cross-validation): Hyperplane Algorithm.

PPCP Algorithm	StatLog Heart Disease - Hyperplane Algorithm					
Error Allowed During	0%,0%	30%,40%	35%,35%	40%,30%	40%,40%	40%,25%
Total: % Correct	55.56	61.76	61.89	64.00	59.09	63.30
Class 1: % Correct	100.00	67.30	61.63	59.37	58.00	63.93
Class 2: % Correct	0.00	54.83	62.21	69.79	60.46	62.50

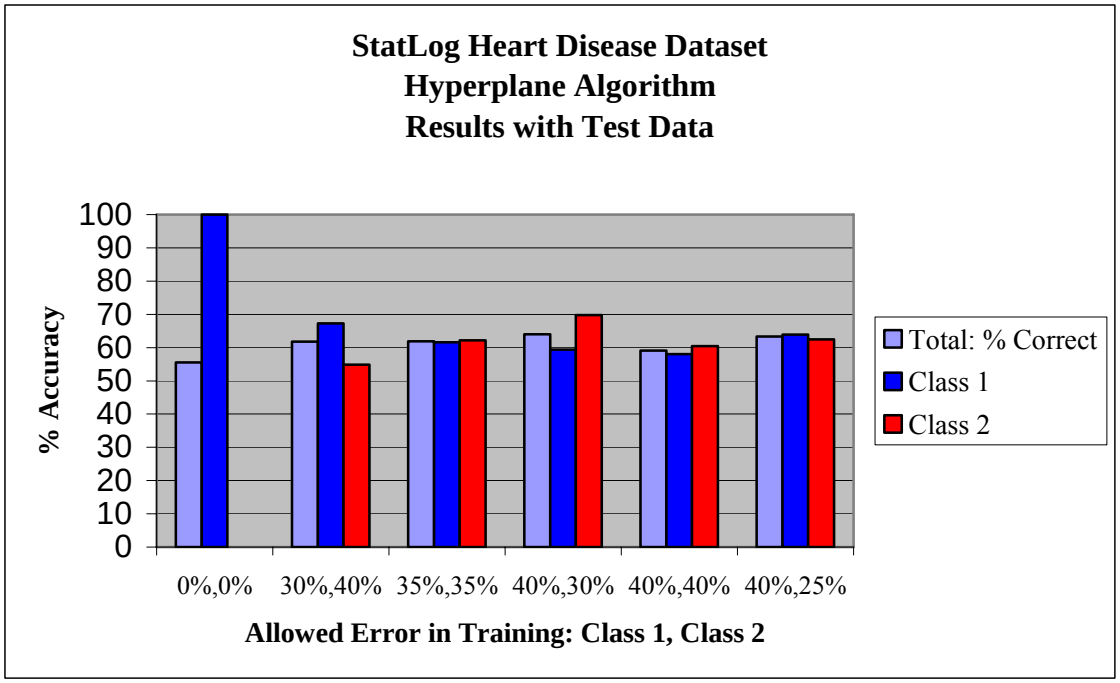


Figure 4-20. StatLog Heart Disease (5-fold cross-validation): Hyperplane Algorithm.

4.4.2 MARGIN ALGORITHM

We present results for each, then compare the two versions.

Local Version

The results (shown in Table 4-29 and Figure 4-21) range from 63.74% average accuracy of classification when the attributes are taken in the order given in the database to 76.98% when rank order by *highest accuracy* is used.

Table 4-29. StatLog Heart Disease (5-fold cross-validation): Margin Algorithm (local).

PPCP Algorithm	StatLog Heart Disease - Margin Algorithm (local)	
	no order	*highest accuracy*
Total: % Correct	63.74	76.98
Class 1: % Correct	57.73	79.00
Class 2: % Correct	71.25	74.46

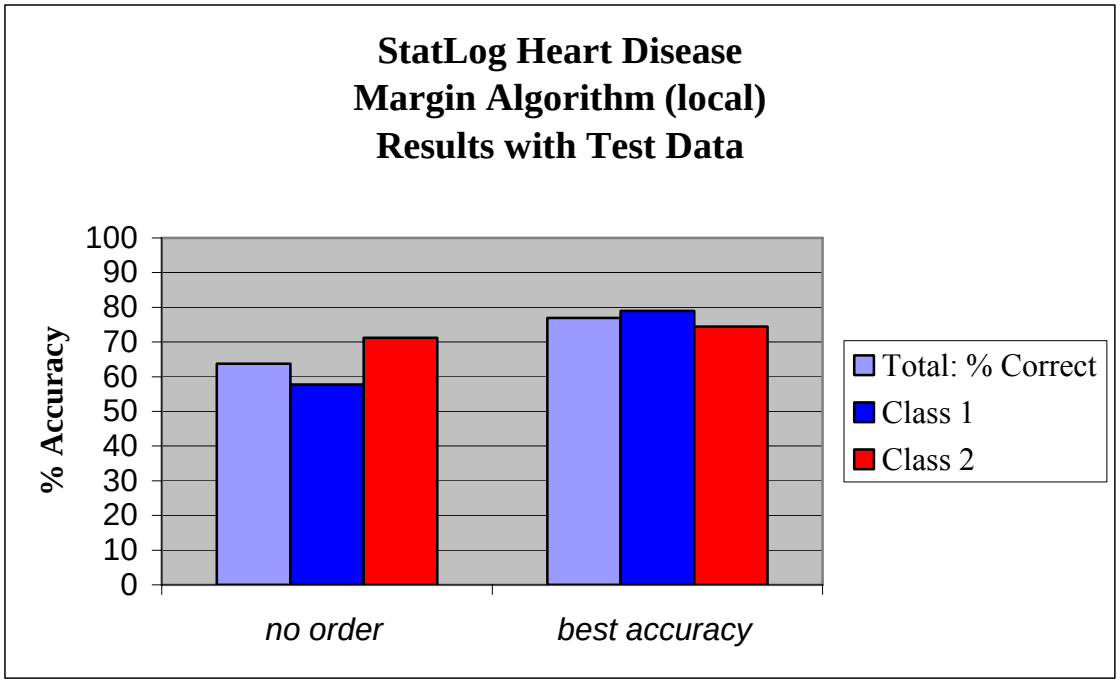


Figure 4-21. StatLog Heart Disease (5-fold cross-validation): Margin Algorithm (local).

Global Version

The results (shown in Table 4-30 and Figure 4-22) range from 63.37% average accuracy of classification when the attributes are taken in the order given in the database to 76.35% when rank order by *highest accuracy* is used.

Table 4-30. StatLog Heart Disease (5-fold cross-validation): Margin Algorithm (global).

PPCP Algorithm	StatLog Heart Disease - Margin Algorithm (global)	
Order of Attributes	no order	*highest accuracy*
Total: % Correct	63.37	76.35
Class 1: % Correct	56.97	79.17
Class 2: % Correct	71.37	72.82

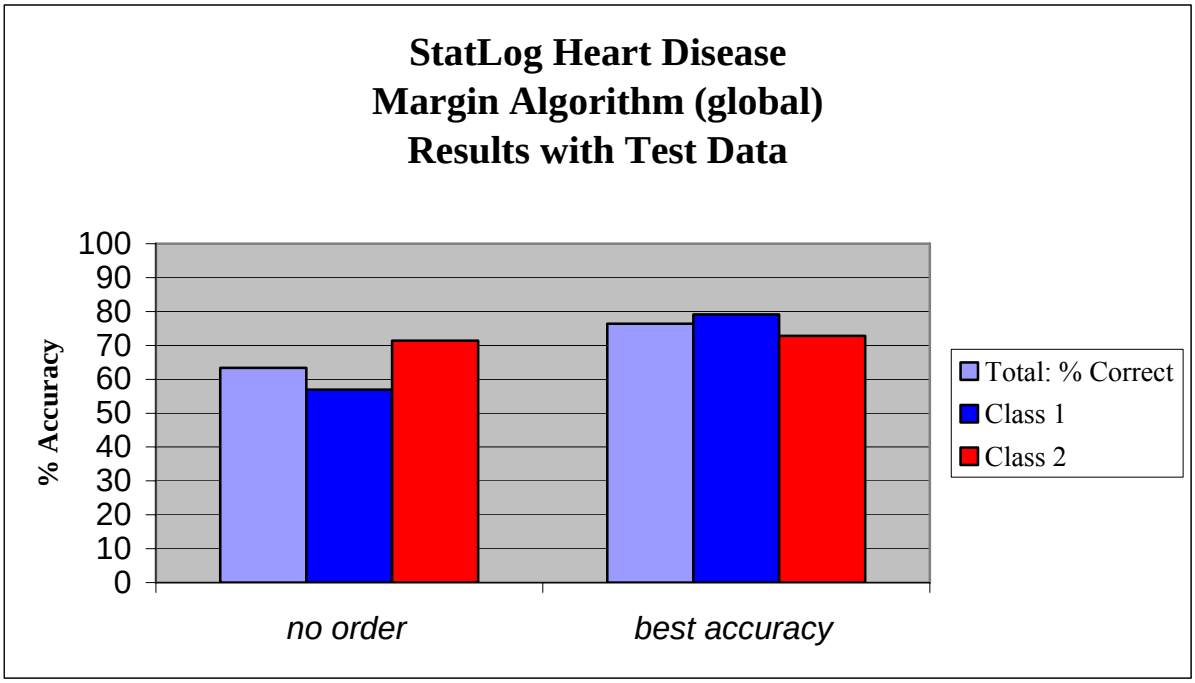


Figure 4-22. StatLog Heart Disease (5-fold cross-validation): Margin Algorithm (global).

Comparison of Versions

In both versions, *Hypothesis 1* is verified. It was expected that this would be more pronounced for the local version, but it was not. Unlike the results in several other databases tested, there appeared to be little difference between versions in the results for overall classification accuracy or for individual classes.

4.4.3 BOX ALGORITHM

Again, a penalty of 0.4 was used in the training as with all other real datasets. With 5-fold cross-validation, *Hypothesis 2* (class order) and *Hypothesis 3* (*hypercubes* versus *symmetric rectangles*) were tested. The results are shown in Table 4-31 and Figure 4-23.

Table 4-31. StatLog Heart Disease (5-fold cross-validation): Box Algorithm (cube).

PPCP Algorithm	StatLog Heart Disease - Box Algorithm (cube)	
Class Order	1,2	2,1
Total: % Correct	66.17	65.93
Class 1: % Correct	88.57	87.60
Class 2: % Correct	38.17	38.83

For the *cube*, as can be seen in the top row of Table 4-31, roughly two thirds of the data is classified correctly regardless of the order of the classes. Also apparent is that the classification accuracies of Class *1* and Class *2* are unaffected by the classification order of classes. This suggests that one class is *not* inside the other, but that there is considerable overlap. The overlap estimated by the Hyperplane Algorithm is 30%+.

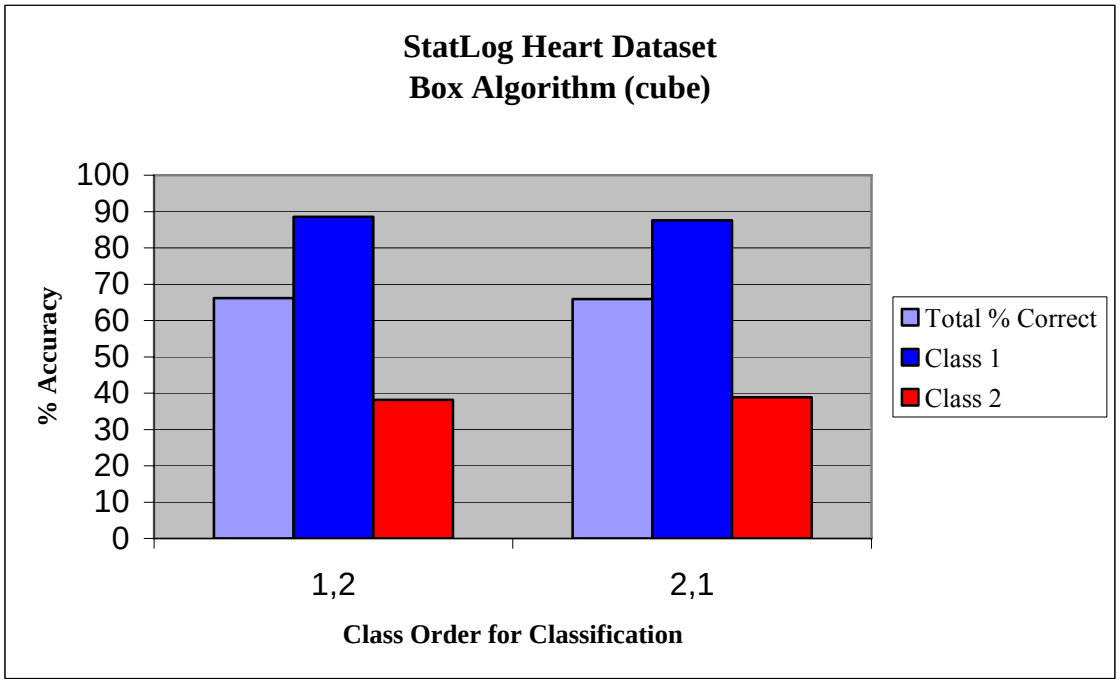


Figure 4-23. StatLog Heart Disease (5-fold cross-validation): Box Algorithm (cube).

For the *symmetric rectangle*, there is an extremely high increase in computational complexity because there are thirteen attributes. Because of this, preliminary testing was used to roughly approximate the limits of the dimensions of the box. As with the Iris data, this was accomplished by using a larger step-size during the *for loops*. We also introduced loop limitations < 3 standard deviations in an effort to make the evaluation possible. Even with these adjustments, an individual run went from seconds or minutes using *hypercubes* to overnight using *symmetric rectangles*.

These results are very preliminary and should not be considered definitive. Each box is determined independently of the others. This would allow parallel computing for such time-consuming instances. The testing portion is 1-2 seconds for either version of the box. Results are shown in Table 4-32 and Figure 4-24.

There is increase in overall accuracy of about 1% - 2% when using the *symmetric box* version. The *symmetric box* version appears to be able to classify individual classes better, as shown in class order 2→1: Class *2* is classified to ~68% accuracy here, while the other class order of the *symmetric rectangle* version and the *cube* version classify Class 2 with 35% - 39% accuracy,. There is a concurrent decrease in classification accuracy for Class *1* of 17%.

Table 4-32. StatLog Heart Disease (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

PPCP Algorithm	StatLog Heart Disease - Box Algorithm (symmetric rectangle)	
Class Order	1,2	2,1
Total: % Correct	67.04	69.63
Class 1: % Correct	92.00	70.67
Class 2: % Correct	35.83	68.3

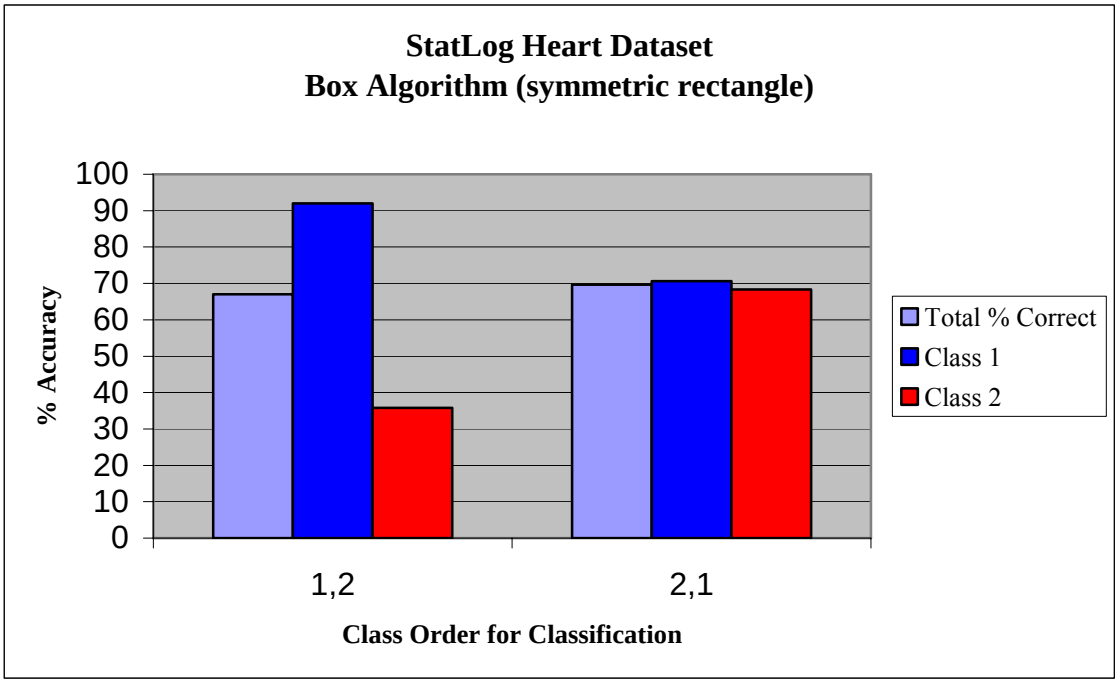


Figure 4-24. StatLog Heart Disease (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

4.4.4 CONCLUSIONS

According to our classification procedure, we expect the Margin Algorithm to give the highest classification accuracy. It does so for this dataset. *Symmetric rectangles* classify more accurately than do *hypercubes*. *Hypothesis 1* and *Hypothesis 3* are verified. *Hypothesis 2* is supported by results for the *symmetric rectangle* but not by the results for the *hypercube.*

4.5 CONTRACEPTIVE METHOD CHOICE

The Contraceptive Method Choice dataset is also a benchmark dataset available from the Information and Computer Science Department at the University of California, Irvine [21]. This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if

they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

The Contraceptive Method Choice dataset has 1473 instances of ten attributes (none missing), plus a class attribute (*no-use*, *long-tern*, or *short-term*). Of the 1473 instances; 629 are *no-use*, 333 are *long-term*, and 511 are *short-term*. The class distribution and a short statistical analysis are shown in Tables 4-33 and 4-34, respectively.

The ten attributes are:

1.	Wife's Age	(numerical)	
2.	Wife's education	(categorical)	1 = low, 2, 3, 4 = high
3.	Husband's education	(categorical)	1 = low, 2, 3, 4 = high
4.	Number of children ever born	(numerical)	
5.	Wife's religion	(binary)	0/1 = Non-Islam/Islam
6.	Wife now working	(binary)	0/1 = Yes/No
7.	Husband's occupation	(categorical)	1, 2, 3, 4
8.	Standard-of-living index	(categorical)	1 = low, 2, 3, 4 = high
9.	Media exposure	(binary)	0/1 = Good/Not good

Table 4-33. StatLog Heart Disease: Class Distribution.

Class Distribution - Contraceptive Method Choice	
Class Value	**Number of instances**
1/No-use	629 (42.7%)
2/Long-term	333 (22.6%)
3/Short-term	511 (34.7%)

Table 4-34. Contraceptive Method Choice: Statistical Analysis.

Brief statistical analysis				
Attribute number	Minimum	Maximum	Mean	Standard Deviation
1	16	49	32.5	8.2
2	1	4	3.0	1.0
3	1	4	3.4	0.8
4	0	16	3.3	2.4
5	0	1	0.9	0.4
6	0	1	0.7	0.4
7	1	4	2.1	0.9
8	1	4	3.1	1.0
9	0	1	0.1	0.3

4.5.1 HYPERPLANE ALGORITHM

The classification accuracy (shown in Table 4-35 and Figure 4-25) when no errors are allowed during training is 41.32%. With various choices of error allowed during training, we did not find a higher accuracy. We suspect that either the classes have a heavy overlap or one class is inside the other class.

Table 4-35. Contraceptive Method Choice (5-fold cross-validation):

Hyperplane Algorithm.

PPCP Algorithm	Contraceptive Method Choice - Hyperplane Algorithm	
Error Allowed During Training: Class 3, Class 1-2; Class 1, Class 2	0%, 0%; 0%, 0%	40%, 50%; 40%, 40%
Total: % Correct	41.32	40.81
Class 1: % Correct	14.60	14.00
Class 2: % Correct	38.99	37.22
Class 3: % Correct	68.29	69.18

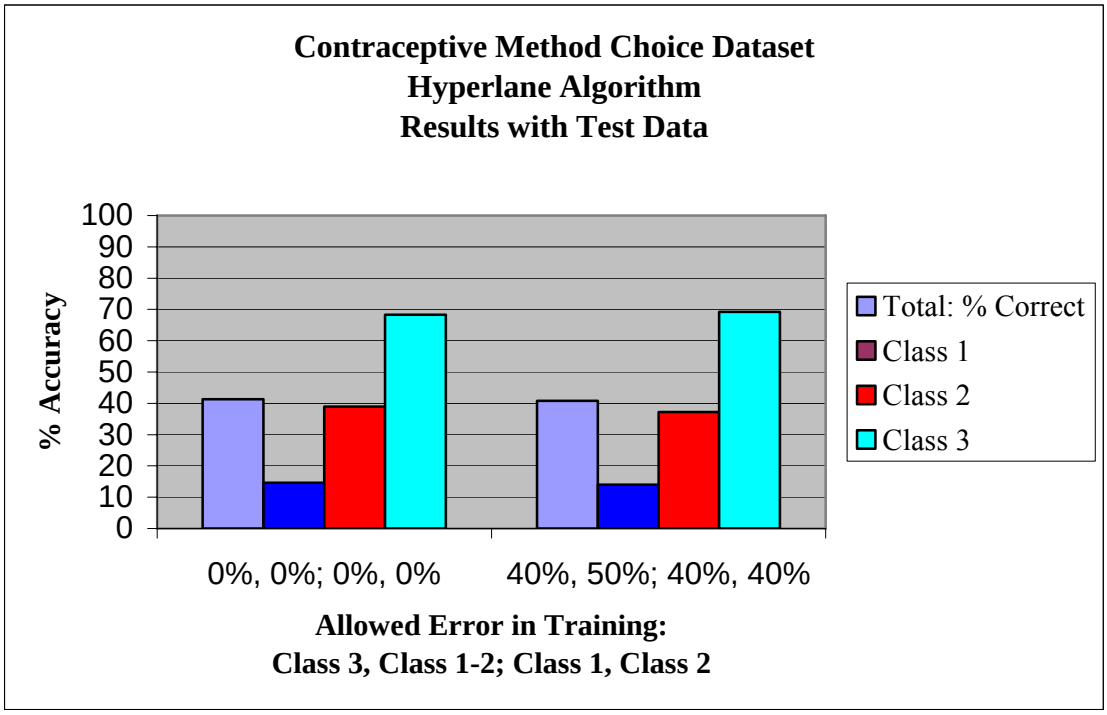


Figure 4-25. Contraceptive Method Choice (5-fold cross-validation): Hyperplane Algorithm.

According to our classification procedure, we would now go to step 3 (the Box Algorithm). Following this, we would decide whether the Hyperplane Algorithm or the Box Algorithm would the classifier of choice. Though the Margin Algorithm would not be used normally, we do so in order to evaluate the classification procedure.

4.5.2 MARGIN ALGORITHM

We present results for each, then compare the two versions.

Local Version

The use of *highest accuracy* does lead to an improvement in classification accuracy, as shown in Table 4-36 and Figure 4-26, but the results are inferior to those obtained by the Hyperplane Algorithm.

Table 4-36. Contraceptive Method Choice (5-fold cross-validation):

Margin Algorithm (local).

PPCP Algorithm	Contraceptive Method Choice - Margin Algorithm (local)	
	no order	*highest accuracy*
Total: % Correct	34.85	38.64
Class 1	0.00	44.55
Class 2	68.76	86.36
Class 3	55.64	0.00

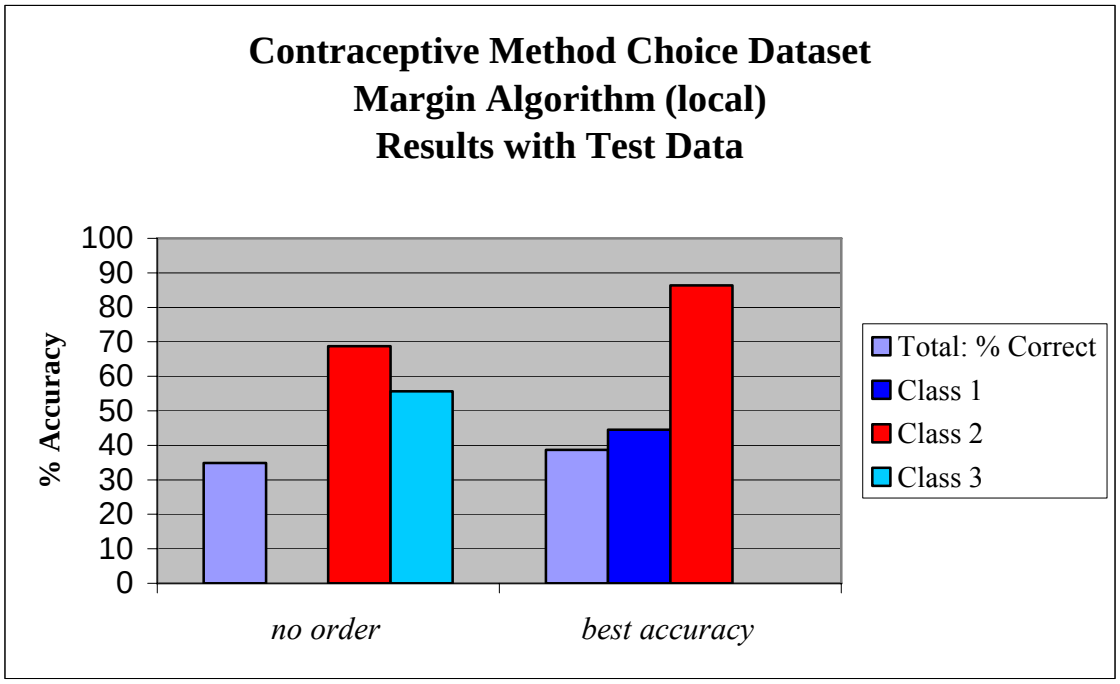


Figure 4-26. Contraceptive Method Choice (5-fold cross-validation):

Margin Algorithm (local).

Global Version

The use of *highest accuracy* does lead to an improvement in classification accuracy for this version as well, as shown in Table 4-37 and Figure 4-27, and the results are somewhat superior (using *highest accuracy*) to those obtained by the Hyperplane Algorithm.

Table 4-37. Contraceptive Method Choice (5-fold cross-validation): Margin Algorithm (global).

PPCP Algorithm	Contraceptive Method Choice - Margin Algorithm (global)	
	no order	*highest accuracy*
Total: % Correct	32.53	42.71
Class 1	1.04	100.00
Class 2	74.19	0.00
Class 3	44.06	0.00

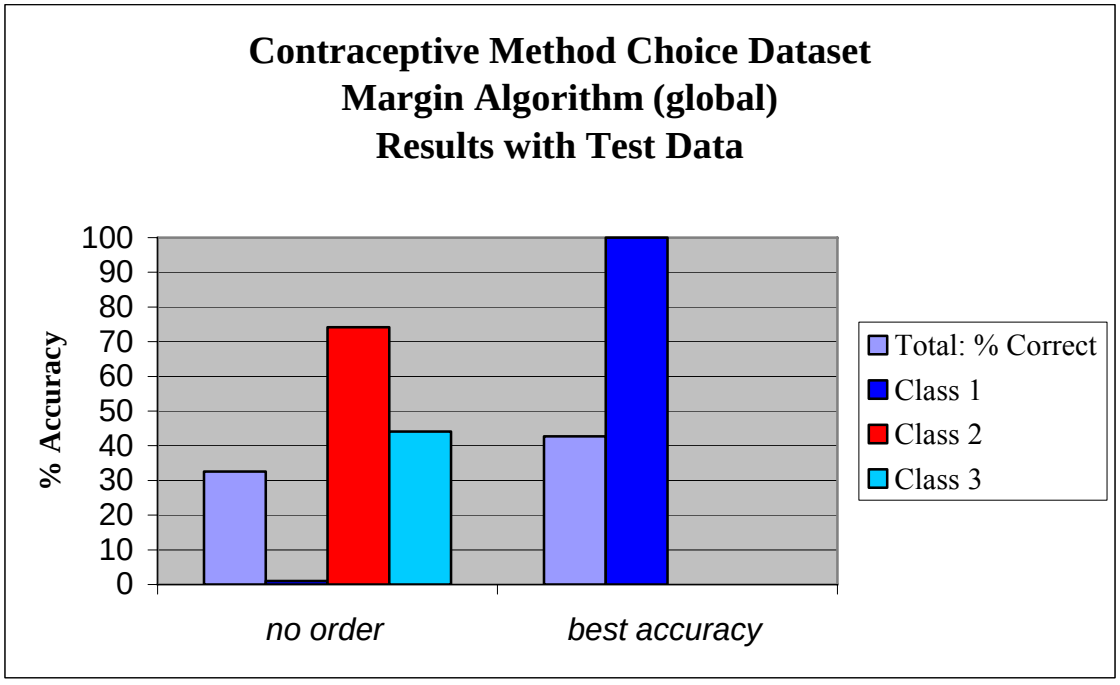


Figure 4-27. Contraceptive Method Choice (5-fold cross-validation): Margin Algorithm (global).

Comparison of Versions

The results show that using either version at least one of the classes is classified completely wrong. We think that this is further evidence of the overlap between classes being extensive. *Hypothesis 1* is supported.

4.5.3 BOX ALGORITHM

A penalty of 0.4 was used in the training as with all other real datasets. Results are shown in Table 4-38 and Figure 4-28.

For the cube, as can be seen in the top row, 40% - 43% of the data is classified correctly regardless of the order of the classes. This corresponds to the majority class. Also apparent is that the class that is used first generally has the best accuracy, with one exception of the six orders. One of the order of classes that uses Class *2* first does best for the entire classification and the other one starting with Class *2* is the only one that classifies each of the three classes with accuracy > 20%.

Table 4-38. Contraceptive Method Choice (5-fold cross-validation):

Box Algorithm (cube).

PPCP Algorithm	Contraceptive Method Choice - Box Algorithm (Cube)					
Class Order	1,2,3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1
Total: % Correct	42.20	42.02	43.26	40.15	40.62	40.66
Class 1 % Correct	98.18	98.62	74.97	24.03	24.44	23.83
Class 2 % Correct	0.18	0.00	50.43	49.40	0.00	2.05
Class 3 % Correct	0.01	0.00	0.00	53.65	87.12	87.02

Class order 2→3→1 is the only time that *all* three classes appear to be modeled at all successfully. Class *1* is classified with ~24% accuracy and the other two classes with ~50% accuracy each.

Inspection of the means for each class reveals that many of the attributes have means that are close to one another. If this were simply one class being inside another, a box inside a box so to speak, the means being close to one another would not matter. We suggest that this is a case of the classes being enmeshed with one another.

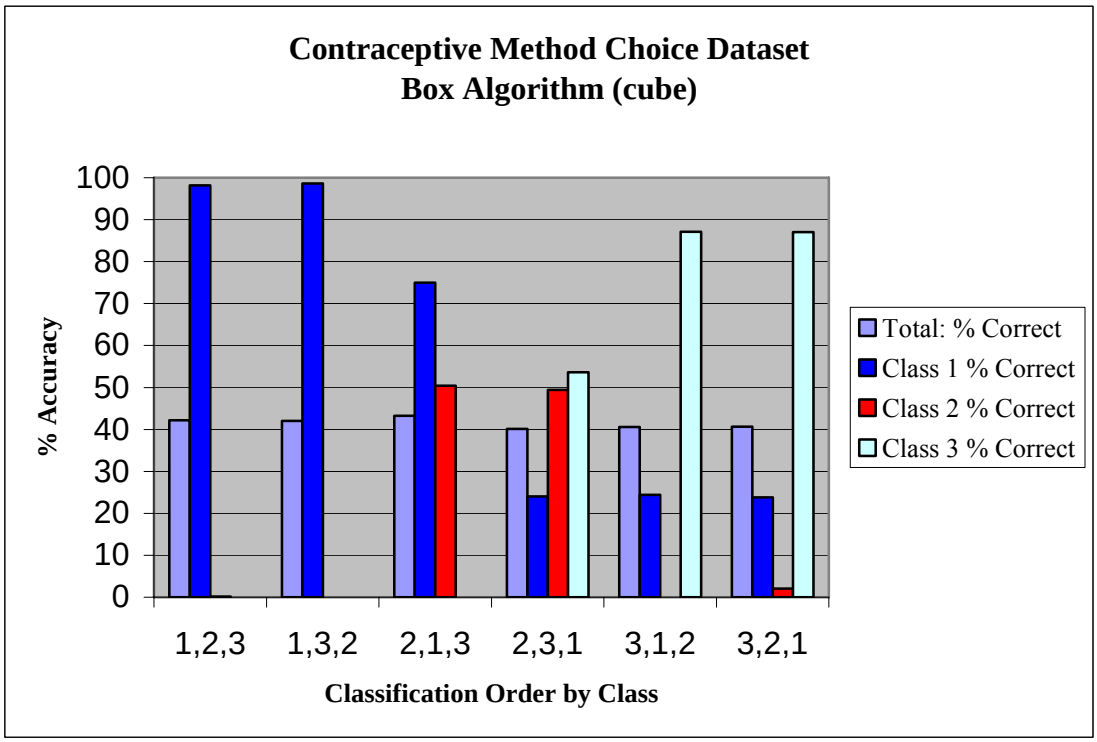


Figure 4-28. Contraceptive Method Choice (5-fold cross-validation): Box Algorithm (cube).

We show results for the *symmetric rectangle* in Table 4-39 and Figure 4-29. Classification accuracy ranges from 41.15% to 49.49%, depending on the order of classes.

Table 4-39. Contraceptive Method Choice (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

PPCP Algorithm	Contraceptive Method Choice - Box Algorithm (Symmetric Rectangle)					
Class Order	1,2,3	1,3,2	2,1,3	2,3,1	3,1,2	3,2,1
Total: % Correct	48.54	41.15	41.15	43.73	49.49	45.08
Class 1 % Correct	48.13	88.26	67.08	34.07	48.42	39.84
Class 2 % Correct	57.10	0.00	50.31	54.76	0.59	10.08
Class 3 % Correct	43.19	5.58	3.29	48.29	83.40	77.46

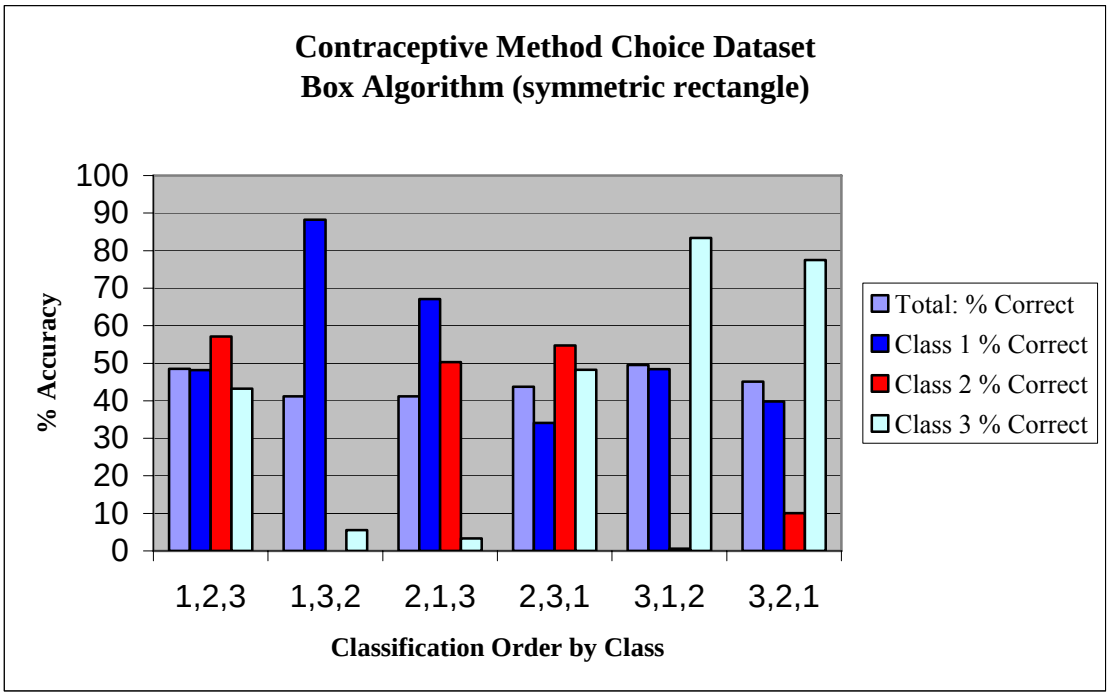


Figure 4-29. Contraceptive Method Choice (5-fold cross-validation):

Box Algorithm (symmetric rectangle).

This dataset has ten attributes. Once again, due to computational complexity for the *symmetric rectangle*, preliminary testing was used to roughly approximate the limits of the dimensions of the boxes. We again implement loop limitations and step-sizes. These results are preliminary and should not be considered definitive.

4.5.4 CONCLUSIONS

According to our classification procedure, we expect the Box Algorithm to give the highest classification accuracy, which it does for this dataset. Using Class *3* first increases the classification accuracy of the entire dataset at the expense of classification accuracy of Class *2*. The order 1→2→3 is the only one that classifies each of the three classes with accuracy > 43%. *Symmetric rectangles* classify more accurately than do *hypercubes*, i.e., an increase up to 6% for overall classification accuracy. *Hypothesis 1*, *Hypothesis 2*, and *Hypothesis 3* are verified.

4.6 OVERFIT

One area of concern for any classification algorithm is *overfit*. In order to evaluate this, we calculate the classification accuracy for the StatLog Heart Disease database using varying amounts of the training set, i.e., from 10% to 90%. As more data becomes available, one expects that the algorithm can do better because there are more examples. *Overfit* when doing this is indicated by an increase then a decrease in classification accuracy on the test set as the percentage of data for training is increased.

This evaluation was carried out for all three steps of the classification procedure and results are shown in Figures 4-30 through 4-32. For this, we split the dataset between training and test sets. When we used 10% was used for training, we used 90% for testing, and so on. The classification accuracies shown are averages for a number of runs: 100 runs except for the *symmetric boxes*, for which we performed ten runs (due to the time required for each run).

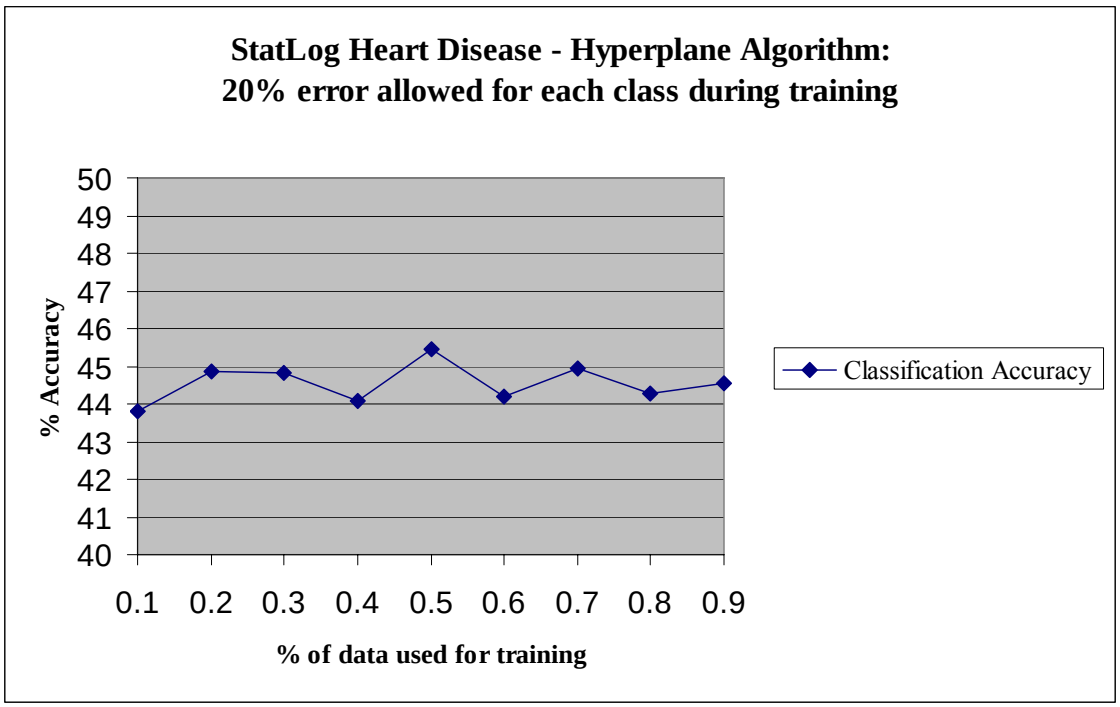


Figure 4-30. StatLog Heart Disease: accuracy vs. percentage of training data (Hyperplane Algorithm).

Note that when the Hyperplane Algorithm is applied (shown in Figure 4-30), the classification accuracy is within a range of 2% for all percentages of training data.

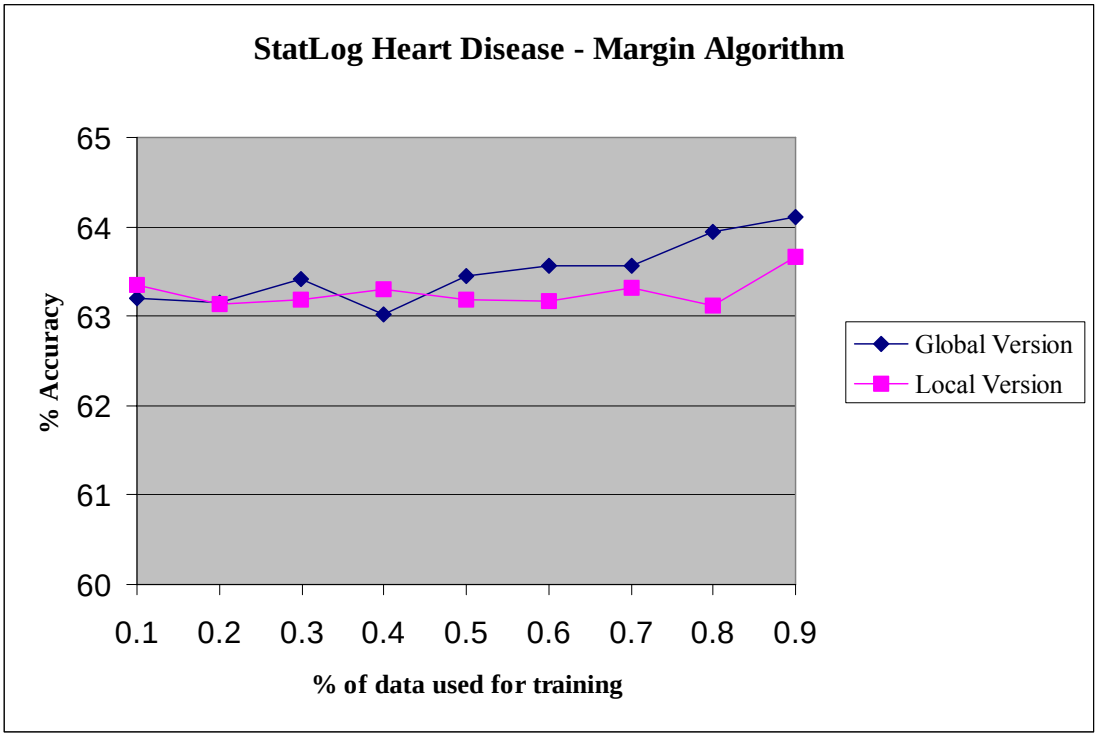


Figure 4-31. StatLog Heart Disease: accuracy vs. percentage of training data (Margin Algorithm).

When the Margin Algorithm is applied (shown in Figure 4-31) and the Margin Algorithm is applied (shown in Figure 4-32), the trend is for the classification accuracy to monotonically increase as the percentage of training data increases.

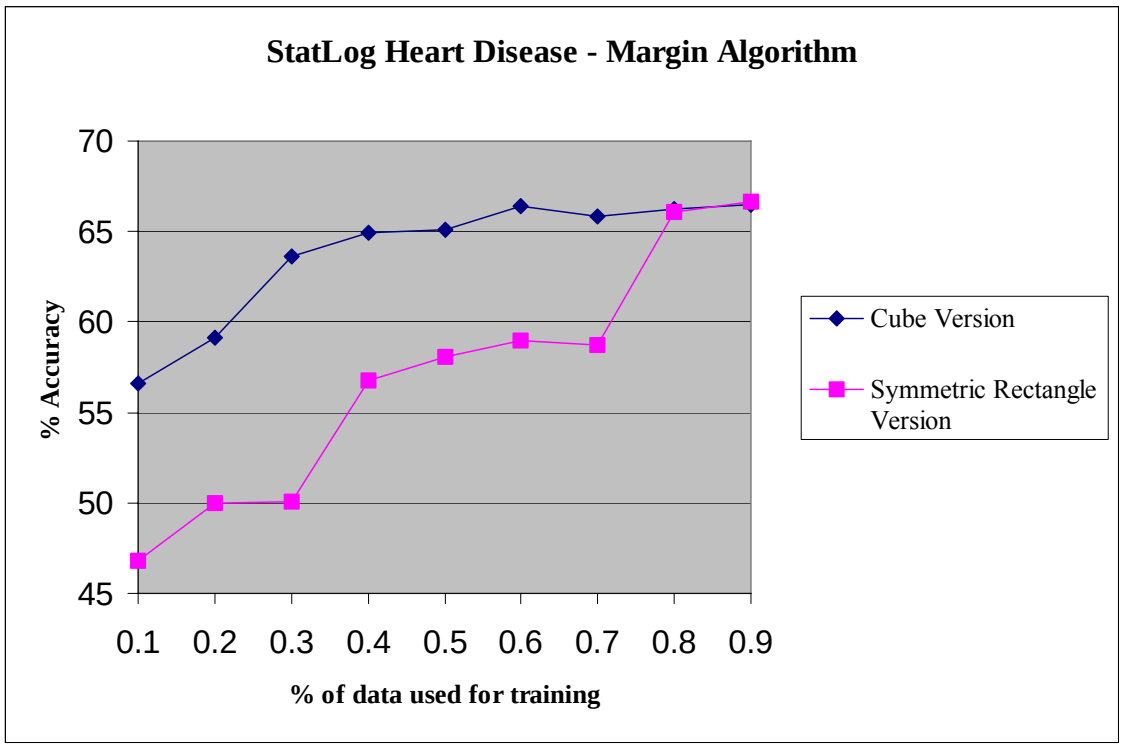


Figure 4-32. StatLog Heart Disease: accuracy vs. percentage of training data (Box Algorithm).

In general, accuracy appears to *not* decrease as it would with *overfit*. Table 4-5, shows, in addition to the learning constants, the classification accuracy for training and testing during ten runs of the global version of the Margin Algorithm for the Wisconsin Breast Cancer dataset. If the data was being overfit, we would expect to see the training data classified with higher accuracy than the testing data for a given run. In fact, this varies – sometimes it is higher, sometimes it is lower – and it is *never* 100%. Figure 4-10 shows the classification accuracy of the local version of the Margin Algorithm on testing versus the size of the training set of the Pima Indians Diabetes dataset. The percentage of the dataset used for training varied from 10% to 70%. If the data was being overfit, we would expect to see the classification accuracy of testing decrease as the percentage of

training data used increased. For the range we tested, the trend was for accuracy to increase as percentage of training data used increased.

These are *snapshots*, i.e., results for specific datasets at specific times. Therefore, the conclusion that overfit is *not* occurring cannot be made. It seems likely that there is much less overfit than with C4.5 though. We suspect that the robust nature of our component algorithms accounts for this:

1. The Hyperplane Algorithm is used with *errors* allowed during training.
2. The Margin Algorithm accepts *errors* in hope of attaining higher accuracy.
3. The Box Algorithm invokes a *penalty* for *errors* but allows them in order to attain a *reasonable box*.

By contrast, univariate decision trees usually classify the training set perfectly, must be pruned [8]. This is because such trees show *overfit* [17]. Results shown for C4.5, as in [14], are for pruned trees.

4.7 CONCLUSIONS

The classification procedure works on the five datasets tested. *Hypothesis 1*, *Hypothesis 2*, and *Hypothesis 3* proved to be true, with exceptions noted in the appropriate sections.

We now compare our results to those of others. In particular, we compare the Hyperplane Algorithm to Support Vector Machines (linear kernel), the Margin Algorithm to C4.5 and CART, and the Box Algorithm to *K*-Nearest Neighbors. All results are summarized in Table 4-40. For each column representing a specific database, the best classification accuracy attained by any algorithm is in bold print. The best classification accuracy attained by our classification procedure is in italicized print.

Where possible, the comparison to Support Vector Machines was restricted to linear kernels. Despite an extensive search of the literature, the comparison for the Contraceptive Method Choice was to SVM with a LS-SVM. LS-SVMs use a least squares approach to overcome computational complexity: *"A modified version of SVM classifiers, Least Squares SVMs (LS-SVMs) classifiers...to obtain a linear set of equations instead of a QP problem in the dual space."* [39].

By [40], we compare to UC0 (CART using 0-SE pruning rules) rather than to UC1 (CART using 1-SE pruning rules). The two versions have a 0.15% – 1.2% difference in classification accuracy. Only in one case was the percentage we reported lower, and then by 0.2%.

The classification procedure performed well on all five datasets. In comparison, SVM had a higher classification accuracy on the StatLog Heart Disease dataset, as did CART and one version of C4.5. The non-linear version of SVM also had a higher classification accuracy on the Contraceptive Method Choice dataset, as did CART. However, across the range of datasets, only CART tied our classification results. We cannot directly compare of the speed of training and testing by PPCP to CART at this time because the results using CART are from different researchers using different measures of speed, when considered at all.

Table 4-40. Summary Chart: Best results for each algorithm.

Algorithm		Wisconsin Breast Cancer	Pima Indians Diabetes	StatLog Heart Disease	Iris	Contraceptive Method Choice
Hyperplane		***96.6%***	69.5%	64.0%	90.2%	41.3%
Margin (global)		94.7%	***78.5%***	*78.8%*	94.4%	42.7%
Margin (local)		93.0%	74.0%	77.0%	*95.6%*	38.6%
Box (cube)		88.15%	65.4%	66.2%	87.7%	43.3%
Box (symmetric rectangle)		92.4%	68.05%	69.6%	90.4%	*49.5%*
Decision Trees	C4.5 R8 (univariate)	94.74% [14]	74.6% [14]	77.0% [14]	95.20% [14]	–
	C4.5 (univariate)	95.75% [40]	73.05% [41] 73.0% [42] 75.8% [40]	80.4% [40]	–	41.7% [40]
	Utree (univariate)	95.7% [43]	72.5% [43]		93.3% [43]	
	CART (multivariate)	95.72–95.88% [44] 93.5% [42] 95.47% [40]	74.48% [41] 72.8% [42] 76.3% [40]	79.3% [40]	94.53% [8]	**54.9%** [40]
Support Vector Machine	SVM (linear)	84.06% [45]	72.37% [45]	**83.86%** [46]	66.43% [45]	–
	LS-SVM (linear)	96% [39]	78% [39]	83% [39]	89.6% [39]	46.9% [39]
K-Nearest Neighbors		75.73% [48] 96.18% [40] 96.2% [43]	67.58% [41] 70.26% [48] 71.9% [42] 70.5% [40] 71.6% [43]	80.42% [48] 77.4% [40]	96.0% [49] **96.7%** [9] 92.7% [43]	40.1% [40]

Chapter 5: Summation

The principal contribution of this thesis is the investigation of classification when using all attributes. The goal was to extend supervised learning to include all attributes in the original feature space without a concurrent increase in computational complexity.

Towards this goal, this thesis proposed dividing it into three regions rather than dividing the original feature space into two regions, as many other algorithms do. It focused on development and evaluation of a classification procedure that employs three separate but related component algorithms, each using the original feature space.

The Hyperplane Algorithm creates hyperplanes reminiscent of Support Vector Machines, but without their computational complexity. The Margin Algorithm is a univariate decision tree that can deal with missing attributes very simply. Brodley, a well-known researcher of multivariate decision trees, states in [43]: *"A strength of univariate decision trees is that they need not evaluate a lot of the input features, which is desirable for representing concepts that are described by a subset of the input features. Indeed for many tasks, the set of relevant features may be unknown and applying a univariate decision tree algorithm to such tasks can generate feedback as to which features are relevant to the task."* Its computational complexity is approximately that of C4.5 [7]. The hypercube version of the Box Algorithm provides a method of classifying classes with large overlap or when one class is enclosed in the other class, while maintaining low computational complexity. The symmetric rectangle of the Box Algorithm, when higher classification accuracy is more important than computational cost, provides a solution to gain it.

5.1 THESIS SYNOPSIS AND CRITIQUE

In Chapter 1, we presented the supervised learning problem, several supervised learning algorithms (Support Vector Machines, decision trees, and K-Nearest Neighbors), and some of the current issues in supervised learning:

1. The Curse of Dimensionality
2. Overfitting
3. Structural Representation Limits

Chapter 2 detailed the classification procedure as well as the three component algorithms. We presented the pseudocode and computational complexity of the algorithm. Each component algorithm was compared to one of the supervised learning algorithms discussed in Chapter 1:

- The Hyperplane Algorithm was compared to Support Vector Machines.
- The Margin Algorithm was compared to decision trees.
- The Box Algorithm was compared to K-Nearest Neighbors.

The evaluation was performed on artificial datasets and results shown in Chapter 3. Heuristics were evaluated for usefulness and several hypotheses tested. It was shown that the *penalty* heuristic derived for the Box Algorithm worked well across all the datasets. The heuristic of *overlap estimation* given by the Hyperplane Algorithm worked well to determine the next step (if needed) of the classification procedure as long as we allowed some small error during training. For these artificial datasets, we can state that it is possible to use all attributes in classification in the original feature space. *Hypothesis 1* was rejected, as it did not give any noticeable improvement. *Hypothesis 2* and

Hypothesis 3 were accepted. From these results, for real-life datasets in Chapter 4, we determined to use:

- The rank order of classification of *highest accuracy* (highest-to-lowest).
- The estimate of overlap attained for the *best* accuracy (with errors allowed during training) to determine the next step of the classification procedure.

In Chapter 4, the practical application of the classification procedure was tested on five real-life datasets and results compared to Support Vector Machines, (univariate and multivariate) decision trees, and *K*-Nearest Neighbors. Again, both the *penalty* heuristic of a particular value for the Box Algorithm worked well across all the datasets and the heuristic of *overlap estimation* given by the Hyperplane Algorithm worked well to determine the next step (if needed) of the classification procedure. In particular,

- All hypotheses for the real datasets are accepted, as shown in detail in Chapter 4. We note that while rank order of classification by *highest accuracy* affect both the global and local versions of the Margin Algorithm, it does *not* always have a greater affect on the local version, as we suspected. The underlying cause was not clear; indicating that more work in this area is needed. We also note that the *symmetric box*es occasionally did not have an improvement in classification error over *hypercubes* for *all* orders of classes tested. We suspect that as the classes are modeled better, the classification order by class is affected; again indicating that more work in this area is needed.
- We show in Table 4-41 that computational complexity can be competitive with other methods. Except for C4.5, PPCP has better time of computation by one or more orders of magnitude. For C4.5, it depends on the values of k and n for the database in question; if k is less than $\log^2 n$, PPCP has a lesser time of computation.

For these real-life datasets, we can state that it is possible to use all attributes in classification in the original feature space. This can be done while maintaining both reasonable computational complexity of training and classification accuracy in testing these algorithms competitive with other algorithms that classify in the original feature space.

Table 4-41. Summary Chart: Computation complexity for each algorithm.

Algorithm	SVM	C4.5	CART	K-NN
Complexity	$O(n^3)$ [11]	$O(k\log n + n\log^2 n)$ [18]	$O(kn^2\log n)$ [1]	$O(kn^2)^*$ [52]
Algorithm	Hyperplane	Margin	Margin	Box (cube)
Complexity	O(kn)	O(kn + klnk)	O(kn + klnk)	O(kn)

- We show in Chapter 4 that *overfit* does not appear to be great for any of the component algorithms. In particular, *overfit* can be reduced with a univariate decision tree, i.e., the Margin Algorithm
- Hyperplane Algorithm:

– The error accepted can be input directly, rather than indirectly as with Support Vector Machines.

- Margin Algorithm, unlike other univariate decision trees:

– Each attribute is tested only once in a path.

– Each attribute is tested in only one subtree.

– Missing values are simply skipped during classification, eliminating the need for *surrogate splits*, etc.

- Box Algorithm:

– The symmetric version can provide increased accuracy.

Classification accuracy for each of the datasets was competitive with existing algorithms that classify in the original feature space and in shown in Table 4-40. Use of the Paired Planes Classification Procedure as an ensemble classifier allows classification on a broader range of datasets than many algorithms.

Two of the components, the Hyperplane Algorithm and the Box Algorithm, require values for every attribute. Real-life datasets often have missing values, thus this is a problem for these two components of the complete classification procedure.

The classification procedure does not work on the XOR problem or with classes in a sinusoidal pattern. This is because when a large percentage of the points are in the margin of overlap for the sinusoidal curve, neither the Hyperplane Algorithm nor the Margin Algorithm classifies any of these points; the Box Algorithm fails completely. When one or both classes are composed of disjoint pieces, it is quite unlikely that the classification procedure would work. The exception is when the disjoint pieces of each class occur in such a way that a hyperplane, margin or box could detect each of the classes without significant overlap of the other class.

5.2 FUTURE WORK

The use of the Hyperplane Algorithm to create approximate distribution curves for each of the classes is seen as a way of visualizing the classes in *k*-dimensional space. According to Duda, "… these theorems highlight the need for insight into proper features and matching the algorithm to the data distribution." [1]. Therefore, these distribution curves may also allow one to better match the problem to the best algorithm. Use of these class distribution curves on an attribute basis may allow a better choice of attribute use in the classification procedure.

Currently, the Margin Algorithm uses the means of the training sets as starting points for determining the parameters. Investigation of alternate starting points, such as the extremes is needed in order to detect boundaries of overlap that the current algorithm does not find.

Speedup of the algorithm is desirable and may be attained by judicious use of the stepsize in the *for loops* for all of the components. One method of doing this is to start with a large stepsize and backup one step at the first unacceptable error, reduce the stepsize, then proceed in a recursive manner.

Appendix A

List of Publications

1. Dan Vance, Anca Ralescu, *Sifting the Margins - An Iterative Empirical Classification Scheme*, Proceedings of 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI), LNAI 3157, Springer-Verlag, 2004, 191-200 (Auckland, New Zealand).
2. Dan Vance, Anca Ralescu, *Sifting the Local Margins: An Iterative Empirical Classification Scheme*, Proceedings of the Fifteen Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), April 16-17, 2004, 131-137 (Chicago, Illinois, U.S.A.).
3. Dan Vance, Anca Ralescu, *A Comparison Between the Local & Global Versions of the Margin Classification Algorithm*, International Conference on Computational Intelligence for Modeling Control and Automation (CIMCA), July 12-14, 2004 (Sydney, Australia).
4. Dan Vance, Anca Ralescu, *Boxes are Better than Circles for Classification*, Proceedings of the 16th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), April 16-17, 2005, 74-82 (Dayton, Ohio, U.S.A.).
5. Dan Vance, Anca Ralescu, *Are Boxes Better for Classification?*, The 48th IEEE International Midwest Symposium on Circuits and Systems, August 7-10, 2005 (Cincinnati, Ohio, U.S.A.) CD: ISBN 0-7803-9198-5, IEEE Catalog Number: 05CH37691C.
6. Dan Vance, Anca Ralescu, *An All-Attributes Approach to Classification*, WCI-NexisLexis Metadata Conference 2005 (Dayton, Ohio, U.S.A.).
7. Dan Vance, Anca Ralescu, *Extending the Margin Algorithm*, International Conference on Computational Intelligence for Modeling Control and Automation (CIMCA), November 27-30, 2005, 389-395 (Vienna, Austria),.
8. Dan Vance, Anca Ralescu, *The Hyperplane Algorithm - A Decision Tree Using Oblique Lines*, Proceedings of the 17th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), April 1-2, 2006**,** Valparaiso University (Chicago, Il, U.S.A.), 2006.

9. Dan Vance, Anca Ralescu, *The Margin Algorithm - A Decision Tree Based on Pairs of Parallel Lines*, 11th Information Processing and Management of Uncertainty International Conference (IPMU), July 2-7, 2006.

Bibliography

[1] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification, 2nd Edition*. John Wiley and Sons, New York. 2001.

[2] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York. 2001.

[3] Robert Schalkoff. *Pattern recognition: statistical, structural, and neural approaches*. John Wiley, New York. 1992.

[4] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, United States of America. 1997.

[5] S.R. Kulkarni, G. Lugosi, and S. S. Venkatesh. *Learning pattern classification a survey*. IEEE Trans. on Information Theory, 44(6): 2178–2206. October 1998.

[6] A.K. Jain, R.P.W. Duin, and J. Mao. *Statistical pattern recognition: a review*. IEEE Trans. on Pattern Analysis and Machine Intelligence, 22(1): 4–37. January 2000.

[7] J.R. Quinlan, Induction of Decision Trees, *Machine Learning* 1: 81-106, 1986. Kluwer Academic Publishers, Boston.

[8] Brodley and Utgoff. COINS Technical Report 92-82, 1992.

[9] A. Lorenz, M. Blum, et al, *Comparison of Different Neuro-Fuzzy Classification Systems for the Detection of Prostate Cancer in Ultrasonic Images*, 1997 IEEE Ultrasonics Symposium Proceedings, 1201-1204, 1997.

[10] O. Mangasarian, N. Street, and W. Wolberg, *Breast Cancer Diagnosis and Prognosis via Linear Programming*, Mathematical Programming Technical Report 94-10 (revised December, 1994), 1994.

[11] Y. Chang, Y. Lee et al. *Data Visualization via Kernel Machines*. Technical Report *C-2006-04*, Institute of Statistical Science, Academia Sinica, to appear in Handbook of Computational Statistics (Volume III) – Data Visualization 2006.

[12] L. Cao and F. Tay. *Support Vector Machine with Adaptive Parameters in Financial Time Series Forecasting*. IEEE Transactions On Neural Networks, vol 14/6, November 2003.

[13] J.R. Quinlan, *C4.5: Programs for Machine Learning*, 1993. San Mateo, CA, Morgan Stanley.

[14] J.R. Quinlan, *Improved Use of Continuous Attributes in C4.5*. Journal of Artificial Intelligence Research 4, 77-90. 1996.

[15] L. Breiman, J. Friedman, et al. *Classification and Regression Trees*. Wadsworth, Inc., U.S.A., 1984.

[16] P. Tan and D. Dowe. *MML Inference of Large Margin Oblique Decision Trees*. Australian Conference on Artificial Intelligence 2004. Springer, Lecture Notes in Computer Science, volume 3339, p1082-1088 2004.

[17] R. Kohavi and J.R. Quinlan, *Decision Tree Discovery* (Chapter 16 of *Handbook of Data Mining and Knowledge Discovery*, 267-276, Oxford University Press, edited by W. Klosgen, J. Zytkow, and J. Zyt. 2002.

[18] E. Frank. *Machine Learning Techniques for Data Mining* (Lecture Notes). Eibe Frank, University of Waikato, New Zealand. 10/25/2000.

[19] H. Abbass, M. Towsey, and G. Finn. *C-Net: A Method for Generating Non-deterministic and Dynamic Multi-variate Decision Trees*, Knowledge and Information Systems: An International Journal (KAIS), Springer, volume 3, number 2, 184-197. 2001.

[20] O.Yildiz, E. Alpaydin. *Omnivariate Decision Trees*, IEEE Transactions on Neural Networks, 12(6), 1539-1546. 2001.

[21] C.J. Merz, P.M. Murphy. *UCI repository of machine learning databases*. University of California, Irvine, Department of Information & Computer Sciences. 1998.

[22] I. Guyon, A. Elisseeff, An Introduction to Variable and Feature Selection, Journal of Machine Learning Research 3 (2003), 1157-1182.

[23] Dan Vance, Anca Ralescu, *Sifting the Margins - An Iterative Empirical Classification Scheme*, Proceedings of 8th Pacific Rim International Conference on Artificial Intelligence (PRICAI), LNAI 3157, Springer-Verlag, 191-200, 2004 (Auckland, New Zealand).

[24] Dan Vance, Anca Ralescu, *Sifting the Local Margins: An Iterative Empirical Classification Scheme*, Proceedings of the Fifteen Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), April 16-17, 2004, 131-137 (Chicago, Illinois, U.S.A.).

[25] Dan Vance, Anca Ralescu, *A Comparison Between the Local & Global Versions of the Margin Classification Algorithm*, International Conference on Computational Intelligence for Modeling Control and Automation (CIMCA), July 12-14, 2004 (Sydney, Australia).

[26] Dan Vance, Anca Ralescu, *Boxes are Better than Circles for Classification*, Proceedings of the 16th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), April 16-17, 2005, 74-82 (Dayton, Ohio, U.S.A.).

[27] Dan Vance, Anca Ralescu, *Are Boxes Better for Classification?*, The 48th IEEE International Midwest Symposium on Circuits and Systems, August 7-10, 2005 (Cincinnati, Ohio, U.S.A.). CD: ISBN 0-7803-9198-5, IEEE Catalog Number: 05CH37691C.

[28] Dan Vance, Anca Ralescu, *An All-Attributes Approach to Classification*, WCI-NexisLexis Metadata Conference 2005 (Dayton, Ohio, U.S.A.).

[29] Dan Vance, Anca Ralescu, *Extending the Margin Algorithm*, International Conference on Computational Intelligence for Modeling Control and Automation (CIMCA), November 27-30, 2005 (Vienna, Austria).

[30] Dan Vance, Anca Ralescu, *The Hyperplane Algorithm - A Decision Tree Using Oblique Lines*, Proceedings of the 17th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS), April 1-2, 2006**,** Valparaiso University (Chicago, Il, U.S.A.), 2006.

[31] Dan Vance, Anca Ralescu, *The Margin Algorithm - A Decision Tree Based on Pairs of Parallel Lines*, 11th Information Processing and Management of Uncertainty International Conference (IPMU), July 2-7, 2006.

[32] N. Christianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Press Syndicate of the University of Cambridge, United Kingdom, 2000.

[33] Online applet for C4.5-type trees, P. Geurts, www.montefiore.ulg.ac.be/~geurts/dtapplet/ dtexplication.html/#online.

[34] F. Provost and T. Fawcett. Robust classification for imprecise environments. Machine Learning, 42(3): 203–210, 2001.

[35] Y. Kawata et al. Computer-aided differential diagnosis of pulmonary nodules based on a hybrid classification approach. Proceedings of the SPIE, 4322: 1796–1806, 2001.

[36] A. Kumar and I. Olmeda. A study of composite or hybrid classifiers for knowledge discovery. INFORMS Journal on Computing, 11(3): 267–277, 1999.

[37] L. Hadjiiski et al. Classification of malignant and benign masses based on hybrid art2lda approach. IEEE Trans. on Medical Imaging, 18(12): 1178–1187, 1999.

[38] I. Olmeda and E. Fernandez. Hybrid classifiers for financial multicriteria decision-making: the case of bankruptcy prediction. Computational Economics, 10(4): 317–335, 1997.

[39] T. Gestel et al. *Benchmarking Least Squares Support Vector Machine Classifiers*. Machine Learning, 54(1), 5-32, January 2004.

[40] T. Lim, W. Loh, and Y. Shih. *An Empirical Comparison of Decision Trees and Other Classification Methods*. Department of Statistics, University of Wisconsin, Madison, Wisconsin, U.S.A. Technical Report 979, June 30, 1997 (revised January 27, 1998).

[41] R. King, C. Feng, and A. Sutherland. *Statlog: Comparison of Classification Algorithms on Large Real-World Problems*. Applied Artificial Intelligence, 9(3): 259-287, 1995.

[42] W. Duch, R. Adamczak, and k. Grbczewski. *A new methodology of extraction, optimization and application of crisp and fuzzy logical rules*. IEEE Transactions on Neural Networks, 11(2), March, 2000.

[43] C. Brodley. *Recursive Automatic Algorithm Selection for Inductive Learning*. Department of Computer Science, University of Massachusetts, Amherst, Massachusetts, U.S.A., COINS Technical Report 94-61, August, 1994.

[44] C. Brodley and P. Utgoff. *Multivariate Decision Trees*. Machine Learning, 19, 45-77, 1995. Kluwer Academic Publishers, Boston.

[45] A. Ali and A. Abraham, *Improved Kernel Learning Using Smoothing Parameter Based Linear Kernel*, IWANN, LNCS 2686, p 206-213, 2003. Springer-Verlag.

[46] C. Park. *Generalization Error Rates for Margin-Based Classifiers*, PhD Dissertation, Ohio State University, 2005.

[47] G. Pandey, H. Gupta and P. Mitra. *Stochastic Scheduling of Active Support Vector Learning Algorithms*, in the Proceedings of the ACM Symposium on Applied Computing (SAC), 38-42, Santa Fe, USA, 2005.

[48] D. Meyer, F. Leisch, and K. Hornik. *Benchmarking Support Vector Machines*. Report 78, November, 2002. Adaptive Information Systems and Modeling in Economics and Management Science.

[49] S. Weiss, C. Kulikowski, *Computer Systems That Learn*, Morgan Kaufman Publishers, Inc, 1991.