Investigating Randomised Sphere Covers in Supervised Learning

Reda Younsi

School of Computing Sciences

University of East Anglia

Thesis submitted to the Postgraduate Research Office in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

February 2011

Abstract

In this thesis, we thoroughly investigate a simple Instance Based Learning (IBL) classifier known as Sphere Cover. We propose a simple Randomized Sphere Cover Classifier (αRSC) and use several datasets in order to evaluate the classification performance of the αRSC classifier. In addition, we analyse the generalization error of the proposed classifier using bias/variance decomposition.

A Sphere Cover Classifier may be described from the compression scheme which stipulates data compression as the reason for high generalization performance. We investigate the compression capacity of αRSC using a sample compression bound. The Compression Scheme prompted us to search new compressibility methods for αRSC. As such, we used a Gaussian kernel to investigate further data compression.

Combining the predictions of a set of classifiers has been vastly successful in classification because of their high classification accuracy. Bagging and Boosting are two popular combination methods known as Meta-learners. That is, they are used to combine predictions of various classifiers. Yet, a large family of IBL classifiers are incapable to use them. We introduce an algorithm that combines several sphere cover classifiers, where each member of the ensemble builds random data-dependent covers. We show this algorithm yield ensembles that are more accurate than a single classifier. We analyse the generalization error of the proposed ensemble using bias/variance decomposition.

We propose a novel subspace method for constructing ensemble of αRSC classifiers. We carry out experiments with several datasets and use the bias/variance decomposition as part of the analysis. We investigate the classifiers proposed in this thesis using three attributes ranking methods on six gene expression datasets, and finally we give a conclusion and discuss future research.

In memory of my father: Si Mouhand-Ourabah.

Acknowledgements

IN THE NAME OF ALLAH, THE MOST GRACIOUS, THE DISPENSER OF GRACE

My thanks and praise goes to my protector, the perfect Allah, who gave me the strength and the ability to finish my thesis in spite of many personal issues.

I would like to thank my supervisor Dr Anthony Bagnall for his very helpful corrections and feedbacks during my research. Dr Bagnall was always there when I needed him and been very patient with my corrections. I would like to thank both my examiners Prof Ludmila Kuncheva and Prof Vic Rayward-Smith for their most precious feedback. I would like to thank Prof Andrew Bangham and all CMP School members, particularly the staff from the Computer and ESCluter services. I would like to particularly thank my Brother Nabil. Finally, I would like to thank all my friends in Norwich and elsewhere for being there when I needed some company.

Contents

List of Figures

List of Tables

LIST OF TABLES

Chapter 1

Introduction

1.1 Learning from Data

The **classification** problem is an important and non-trivial problem in machine learning research. It is important because data is now generated and stored in huge quantities, requiring researchers to build decision making machines that are both efficient and effective. In classification, this machine is presented with **labelled examples** and is required to learn to differentiate between them. **Learning**, in this case, means finding a way to represent the examples as input and their categories as labels, and to generates a **hypothesis** or a **classifier** that maps inputs to desired outputs. In addition, it is required for this machine accurately predict unseen examples. This is a fundamental problem in classification which is known as **generalisation**. The machine is called a **learning algorithm** and the field is known as **supervised learning**.

In this section we define the classification problem from a mathematical perspective and give some important concepts used in supervised learning research. We first present the general notion of learning in classification problem. Let $\mathcal{X}$ be the set of all examples $\mathbf{x}$, which we might call the **input data**. Let C be a finite set of target classes that we call the **output**. The classification for a binary class problem is to learn some decision rule distinguishing between objects belonging to one of two classes, based on a set of m training examples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$, $\mathbf{x}_i \in \mathcal{X} \subset \mathbb{R}^d$, d is the number of attributes, $y_i \in \{1, -1\}$, where $\mathbf{x}_i$ represents a vector of measurements describing the i^{th} example, and y_i indicates the class to which the i^{th} example belongs, with $y_i = +1$ representing class C_1 and $y_i = -1$ representing class C_2. An important assumption to make about D is that the sample is drawn identically and independently from an unknown but fixed probability distribution P_D.

In multi-class classification setting we are given n training examples $\mathbf{x} = (\mathbf{x}_1, ..., \mathbf{x}_n) \in \mathcal{X}$ with the corresponding **class labels** $\boldsymbol{y} = (y_1, ..., y_n) \in C$. This training sample is defined as $(\mathbf{x}, y) = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)) \in (\mathcal{X} \times C)$. The learning problem in classification is defined as:

Definition 1.1.1 (Learning problem)[56] *The learning is to find the unknown (functional) relationship* $f \in \mathbb{R}$ *between example* $\mathbf{x} \in \mathcal{X}$ *and target* $y \in C$ *based solely on a sample* $(\mathbf{x}, y) = ((\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)) \in (\mathcal{X} \times C)^n$ *of size* $n \in \mathbb{N}$ *drawn iid from an unknown distribution* $P_{\mathcal{X}C}$*.*

Definition 1.1.1 describes the learning of a function on the sample space, which is also called the hypothesis, and gives a value for each point in the sample space.

As an example we are given a number of water samples from different wells for analysis. After producing a bacteriological and chemical analysis of the samples each tube is labelled with +1 if drinkable and -1 otherwise, that is $y = \{-1, +1\}$ is the class label. The analysis results are stored in a database, such that future water samples are checked against the values in the database to decide whether the water is drinkable or not. Let suppose that d analysis were made and each result is a real value describing the quantity of a given bacterial or chemical substance in the water. Each water sample is then represented by a vector $\mathbf{x}_i = \{a_1, ..., a_d\}$ which we call the **attribute vector**. The collected database is divided into a **training set** that we can use to train a learning algorithm. We normally select a small subset of these training set for the **validation** task, this set is called the **validation set**. The remaining part of the database called the **test set** is used to predict the label, hence the performance of the classifier.

A learning algorithm is defined as:

Definition 1.1.2 (Learning algorithm)[56] *Given an example space* $\mathcal{X}$*, an output space* C *and a fixed set* $F \subset \mathbb{R}^{\mathcal{X}}$ *of functions mapping* $\mathcal{X}$ *to* $\mathbb{R}$*, a learning algorithm* $\mathcal{A}$ *for the hypothesis space* F *is a mapping*

$$\mathcal{A} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times C)^n \to F$$

Assuming that we have an infinity of examples to choose from then the learning algorithm $\mathcal{A}$ of definition 1.1.2 finds all the mappings (functions or hypotheses) in the fixed set F. If we know the probability distribution P_D, the classifier that minimizes the misclassification risk given this probability distribution is the Bayes classifier, and in this case, the classification problem becomes trivial. However, as our water example shows, we only have access to a finite number of samples to use in learning the hypothesis. We evaluate the performance of the hypothesis with some mathematical quantities such as the empirical risk as defined below. The loss function for the classification problem is defined as:

$$L_{0/1}(f(\mathbf{x}), y) \overset{def}{=} \mathsf{I}_{f(\mathbf{x}) \neq y}$$

I is the identity function with $\mathsf{I}(\text{true}) = 1$ and $\mathsf{I}(\text{false}) = 0$,

$L(f(\mathbf{x}), y)$ being the 0/1 loss function which is a measure of cost when making a prediction at an example $\mathbf{x}$ is $f(\mathbf{x})$ but the true class is y. The $L_{0/1}$ function is the most commonly used type of loss function in classification. Therefore, the empirical risk R_{emp} of a function f, $f \in F \subseteq \mathbb{R}$, F being the **function class**, and given a training set D is:

$$R_{emp}[f, D] \overset{def}{=} \frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}_i), y_i),$$

It is well known that if the function class is too complex then **overfitting** happens which results in a depreciation of the expected risk $R[f]$ defined as:

$$R[f] \overset{def}{=} \mathsf{E}_{\mathbf{x}}[L(f(\mathbf{x}), y)],$$

The above defines the mathematical quantities for the classification problem. The main task in classification is to find the hypothesis that minimizes the expected or true risk over the given samples. We need to use the empirical risk to assess the quality of this hypothesis then hope that this extrapolates to the true risk.

1.2 Thesis Objectives

1. To propose a simple Randomized Sphere Cover Classifier and to explore it as a base classifier for ensemble methods.

2. To examine pruning (i.e. removing spheres using a threshold value) as α regularization parameter to penalize complex covers, and to investigate whether pruning low cardinality spheres improves the generalization performance of unpruned Randomized Sphere Cover Classifier.

3. To compare the accuracy of the Randomized Sphere Cover Classifier with some of the more commonly used classifiers.

4. To investigate the contribution of bias and the variance to the prediction error in the Randomized Sphere Cover Classifier.

5. To examine the complexity-accuracy trade-off of the Randomized Sphere Cover Classifier using compression scheme [38, 82, 134].

6. To investigate whether using the kernel method will reduce further the number of prototypes (spheres) and whether the generalization error of the kernel Randomized Sphere Cover Classifier will decrease.

7. To investigate a new combination method for the Randomized Class Cover Classifiers based on sampling. This combination method uses a new parameter as part of perturbing covers for generating diversity.

8. To investigate the bias and variance decomposition of the proposed ensemble and compares the results with those of a single Randomized Sphere Cover Classifier.

9. To investigate the proposed ensemble in the subspaces. This is done using random attribute subsets to build covers for the ensemble. It also investigate the contribution of the bias and the variance to the prediction error.

10. To examine the usefulness of the proposed ensembles on real world gene expression datasets.

1.3 Thesis Organization

Chapter 2 Introduce various ensemble methods, and describe the bias variance decomposition as an analytical tool for the proposed algorithms.

Chapter 3 Describes the class cover problem as the source for various class cover algorithms in which the Randomized Sphere Cover Classifier is derived. A quantitative analysis of a Randomized Sphere Cover Classifier is carried out. This chapter also considers the bias/variance decomposition to study the effect pruning has in the prediction error.

Chapter 4 Considers the compression scheme as a new tool to analyse sphere cover algorithms and proposes a new sphere cover classifier using Kernel method.

Chapter 5 Describes a new ensemble based on Randomized Sphere Cover Classifiers and studies its performance using bias/variance decomposition. It also describes a new subspace method based on the ensemble and examines its performance using bias/variance decomposition.

Chapter 6 Evaluates the proposed classifiers on gene expression datasets and study its performance using three attribute ranking methods.

Chapter 7 Conclusions and Future research.

1.4 Benchmark Datasets

To evaluate the performance of the proposed classifiers, we used twenty four datasets from both UCI data repository [39], and boosting repository (http://ida.first.gmd.de/raetsch/data/benchmarks.htm). These datasets are summarized in table 1.1. They were selected because they vary in the numbers of training examples, classes and attributes and thus provide a diverse testbed. In addition, they all have only continuous attributes.

Table 1.1: Benchmark datasets used for the empirical evaluations

Dataset	Examples	Attributes	Classes	Dataset	Examples	Attributes	Classes
Sonar	208	60	2	Vehicle	846	18	4
Glass6	214	9	6	Vowel	990	10	11
Glass2	214	9	2	German	1000	20	2
Thyroid	215	5	2	Concentric	2000	2	2
Heart	270	13	2	Image	2310	18	2
Haberman	306	3	2	Abalone	4177	8	3
Cancer	315	13	2	Clouds	5000	2	2
Ecoli	336	7	8	Waveforme	5000	40	3
Ionosphere	351	34	2	Ringnorm	7400	20	2
wdbc	569	30	2	Twonorm	7400	20	2
Winsconsin	699	9	2	Pendigitis	10991	14	10
Diabetes	768	8	2	Magic	19020	2	10
Yeast	1484	8	10	Satimage	6435	36	6

1.5 Software Package used in this Thesis

- The sphere cover algorithms were implemented in C++.

- The CCCD program described in by Marchette [1] written in R then integrate it in our C++ program.
- The bias/variance decomposition C++ code of Domingos [34] was integrated with our program.
- We build the compression bound in C++ using the approximation (Stirling's Series) that is used to compute the binomial coefficients in the multiple bounds program written in C++ by Kääriäinen and Langford [69].
- WEKA software was used to compare our results with those of other learning algorithms.
- We used gist toolbox for the kernel transformation [2]. The transformed distance matrix is then used with our program.
- We run the entire experiments using Cluster1 and EScluster provided by the EScience department at UEA.

[1]http://cran.r-project.org/web/packages/cccd/
[2]http://bioinformatics.ubc.ca/gist/index.html

Chapter 2

Background

A vast number of classification algorithms are commonly used for many applications. A full description of these algorithms and their applications can be found in Maimon and Rokach data mining book [86]. In this section, we give a general description of several IBL classifiers in order to make a direct connection to classifiers we describe in chapter 3. In addition, we review similar sphere cover algorithms using different architectures

2.1 Instance Based Classifiers

2.1.1 The (K) Nearest Neighbour Algorithm

The Nearest Neighbour (NN) classifier has been described as the "simplest and most intuitive pattern recognition paradigm" [79]. NN uses the Euclidean distance function to classify an example according to the class of its nearest neighbour in a training set. This choice of distance metric is straightforward for continuous attributes although scale and variance are obviously issues. However, nominal and mixed attributes can be a problem for NN classifier which requires the modification of the distance metric [1]. The classification rule of the NN classifier is depicted in figure 2.1 using a Voronoi diagram. In Voronoi diagrams the Euclidean space is decomposed into regions around each example, such that all the examples in the region around a given examples $\mathbf{x}$ are closer to $\mathbf{x}$ than any other examples. Finding the nearest neighbour of $\mathbf{x}$ is equivalent to determining which cell in the Voronoi diagram contains $\mathbf{x}$. The K nearest neighbour algorithm (K-NN) considers K training examples nearest to a test example and classifies it as the class label of the majority. K-NN was shown to be more robust to noisy datasets in comparison to the NN classifier [140].

The primary problem with NN classifier is the computational and data storage load, since a distance matrix of the entire training set is calculated. kd-tree is a binary tree that partitions the data into rectangular regions in order to facilitate the search to the closest example [1]. A possible solution to data storage is to reduce the number of training examples. This has been a major focus of attention in the machine learning literature [17, 70, 142].

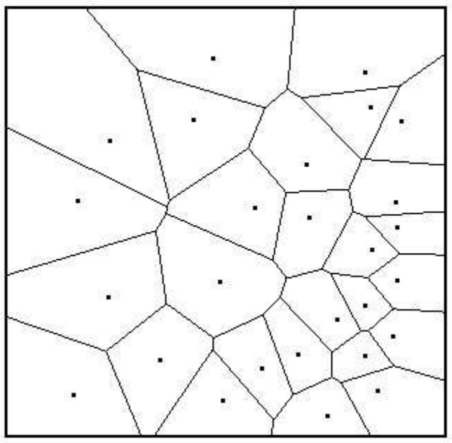


Figure 2.1: A Voronoi diagram for the nearest neighbour showing piecewise linear decision boundaries. Black dots represent examples and cells delimit its neighbourhood

We can review two techniques that remove examples from the training set: editing and condensing [79]. The outcome of editing methods on noisy datasets is smooth decision boundaries since they concentrate on close border cases, eliminating a possible overlap between class regions [2]. Alternatively, condensing methods select a small subset of examples without a significant degradation in the classification accuracy [79]. The main concern of data reduction classifiers is to search and select the best set of examples for each class to keep for classification. Examples retained for classification are called **prototypes**. Using multiple prototypes allows defining a variety of class regions shapes making prototype-based (also known as instance-based and exemplar-based) methods powerful classifiers [10, 11, 79].

2.1.2 Nested Generalised Exemplars Algorithm

The Nested Generalised Exemplars algorithm (NGE) is an instance-based learning classifiers that uses hyper-rectangles [115]. The main idea is to cover examples using hyper-rectangles allowing these to nest or overlap. Hyper-rectangles are represented as a class value and the bounds on each attribute that define its borders. For continuous attributes, the maximum and minimum attribute-values are stored. These maximum and minimum values describe the range of values covered by the hyper-rectangle. In general, hyper-rectangle based algorithms are prone to overfitting [139]. Wettschereck and Dietterich [139] showed that NGE performed badly using various datasets blaming nesting and overlapping as the main cause. Wettschereck [137] attempt to solve this using a hybrid of Nearest-Neighbor and Hyper-rectangle Algorithm. In the other hand, Martin [92] used an algorithm that disallows overlaps by pruning hyper-rectangles that conflicted with a new examples. The non-nested Generalised exemplars (NNGE) extend the boundary of existing hyper-rectangles for each new data. For continuous attributes, the maximum value is increased, or minimum decreased until the new example is covered by the hyper-rectangle. The performance of this algorithm matches that of the decision tree C4.5 [108] but does poorly on noisy datasets [92]. Recently, the NNGE algorithm has been revisited using an improved classification rule [45] which showed better performance in comparison to previous results. A rectangle based classifier with its respective decision boundaries is shown in figure 2.2.

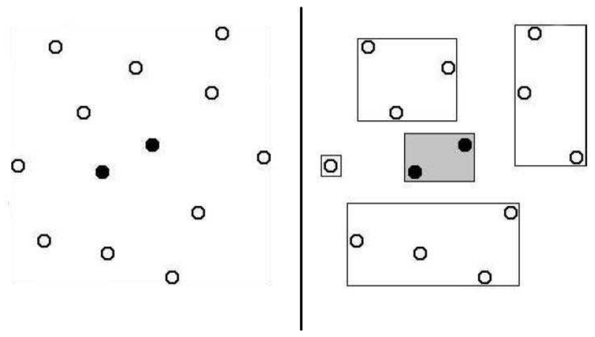


Figure 2.2: A NNGE classifier on a binary toy data (○ = class1 , ● = class 2). Rectangles represent the geometrical generalization of the input space.

2.1.3 The Class Cover Problem (CCP)

In this section, we discuss the CCP which is the source of a variety of sphere cover algorithms used for classification, notably the Class Cover Catch Digraph (CCCD). The CCP is to find the smallest set of **covering spheres**, the union of these spheres are termed **a cover**. The class cover problem (CCP) was first introduced in [23]. For a given class C_1, a sphere B_i, with centre $\mathbf{c}_i$ and radius r_i is defined as the set of data $B_i = \{\mathbf{x} \in \mathcal{X} : \mathfrak{d}(\mathbf{x}, c_i) < r_i\}$. A dissimilarity measure $\mathfrak{d}$ on $\mathcal{X}$ is a function $\mathfrak{d} : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}_+$ such that $\forall \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$, $\mathfrak{d}(\mathbf{x}_1, \mathbf{x}_1) = 0$ and $\mathfrak{d}(\mathbf{x}_1, \mathbf{x}_2) = \mathfrak{d}(\mathbf{x}_2, \mathbf{x}_1) \geq 0$. Therefore, a sphere B_i covers a number of examples $\mathbf{x}_i$ from only one class C_1. The radius of the sphere B_i is defined as the distance from the centre to the closest example from class C_2, $\mathbf{x}_j$ i.e. $r_i = \mathfrak{d}(\mathbf{c}_i, \mathbf{x}_j)$ where $\mathbf{x}_j \in C_2$ such that $\mathfrak{d}(\mathbf{c}_i, \mathbf{x}_j) \leq \mathfrak{d}(\mathbf{c}_i, \mathbf{x}_k)$, $\forall \mathbf{x}_k = C_2$ s.t $y_k = C_2$. The union of the spheres that contain all of the examples of one class and does not contain any of the other class is called a **pure and proper cover**. The class cover problem for two class problems is to find the pure and proper cover for one class with the minimum number of spheres. Each class is dealt with separately. The CCP can be easily extended to multi-class problem.

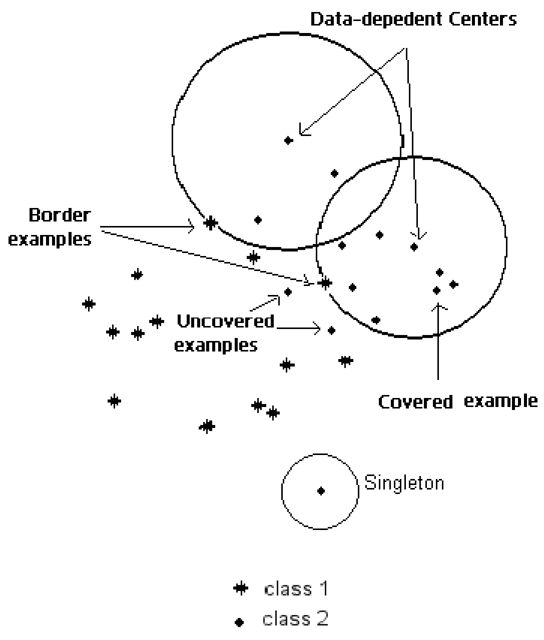


Figure 2.3: A data dependent Class Cover terminology which shows a pure and no-proper cover. Pure because no examples from different class are allowed in the cover and no-proper because two example are uncovered. The singleton circle is defined as the circle that has only a data example for its centre.

A number of variations of the CCP are possible. The CCP that uses spheres that have centres from the training set and have the same radius is called **constrained** and **homogeneous** CCP [107]. The CCP that uses spheres that have centres from the training set but can have different radii is called constrained and **inhomogeneous** CCP. The constrained CCP is also referred to as **data-dependent** CCP [21]. The CCP can easily be turned into a classifier by observing whether an example is interior to one of the spheres of a given class. Various modifications to this simple class cover classifier have been explored in [107]. A CCP classifier is related to IBL classifiers in that they choose a subset as representatives or prototypes.

2.1.3.1 The Class Cover Catch Digraph

The class cover of a training examples $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{l}$ is normally defined by the collection of all the spheres that cover class C_1 but none of the class C_2 such that for each example $\mathbf{x}_i \in D$ there is a sphere B_i centred at $\mathbf{x}_i$. This is a highly redundant cover which requires a reduction in the number of spheres, and the CCP involves finding the optimal sphere reduction.

Selecting the smallest subset of spheres whilst preserving the class cover improves classification by avoiding overfitting [21]. The standard approach to solving the CCP is to consider it as a graph-theoretic problem [30, 31]. Priebe *et al* [29] proposed the Class Cover Catch Digraph (CCCD). A CCCD is a special type of **directed graph** or **digraph** that has several properties which could be useful for the class cover problem [30].

A digraph $D = (V, E)$ consists of a vertex set $V = \{v_1, ..., v_n\}$ and an edge set $E = \{\{v_i, v_j\}\} \subset V \times V$. $e = (v, w)$ denotes and edge between v and w if there is a unique edge e associated with the ordered pair (v, w) of vertices. A directed graph is shown in Figure 2.4. The directed edges are indicated by arrows. Edge e_4 is associated with the ordered pair (v_3, v_4)

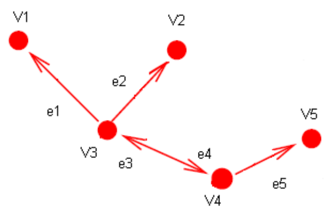


Figure 2.4: Directed graph (digraph) with vertex set $V = \{v_1, v_2, v_3, v_4\}$ and edges set $E = \{e_1, e_2, e_3, e_4, e_5\}$ represented by arrows.

of vertices, and edge e_1 is associated with the ordered pair (v_3, v_1).

A **Euclidean Class Cover Catch Digraph** is a CCCD where the sets are spheres and the examples are the sphere centres (also called a **sphere diagraph**). The spheres define regions of neighbourhood and examples inside these regions are connected via edges. The

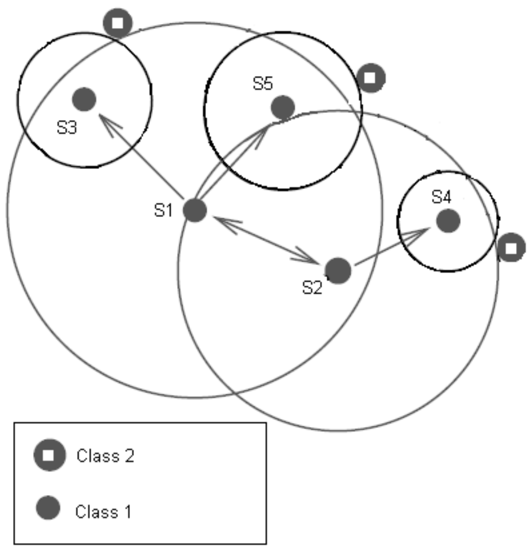


Figure 2.5: A sphere digraph showing the sphere set $S = \{S1, S2, S3, S4, S5\}$ and edges represented by arrows. The edges are connect only if the vertex is inside a sphere. The vertex of the sphere S1 is linked to vertices S2, S3, and S5. The vertex of sphere S2 is linked to vertices S1 and S4. Accordingly, this sphere digraph shows that sphere S3, S4 and S5 are redundant.

examples corresponding to the centres will be from a different class than those that define the radii. This will allow the building of sphere digraphs whose underlying spheres cover the observations from one of the classes, at the exclusion of the examples from the other class. An example of a sphere digraph for a two class toy problem is shown in Figure 2.5.

The **open neighbourhood** of a vertex $S1 \in V$ of Figure 2.5, denoted $N(S1)$, is the set of vertices with edges from $S1$, thus $N(S1) = \{w \in V \setminus \{S1, w\} \in E\}$ which are $\{S2, S3, S5\}$. The **closed neighbourhood** is $N[S1] = N(S1) \cup \{S1\}$. Thus, the closed neighbourhood of the sphere $S1$ is the set of spheres $\{S1, S2, S3, S5\}$.

A **dominating set** of a sphere digraph is a subset $\mathcal{D}$ of the vertices such that every vertex $v \in V$ is either in $\mathcal{D}$ or there exits a $w \in \mathcal{D}$ with $\{w, v\} \in \mathcal{D}$. It is said that such vertex is dominated by w. Thus, $\mathcal{D}$ is any set of vertices such that $\bigcup_{w \in \mathcal{D}} N[w] = V$. The domination number λ is the number of vertices in $\mathcal{D}$. The CCCD dominating set with minimum cardinality of Figure 2.6 is the set $\mathcal{D} = \{S1, S2\}$. The decision surface of a CCCD, which is the surface used by a CCCD for classification, is shown in Figure 2.7

Finding a minimum dominating set of a CCCD is NP-Hard [21, 22]. An approximation

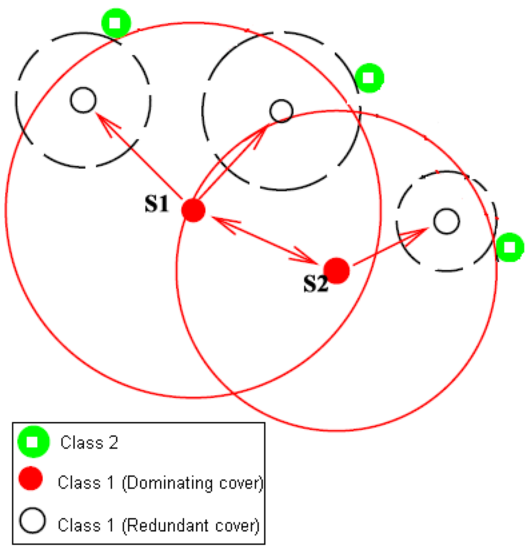
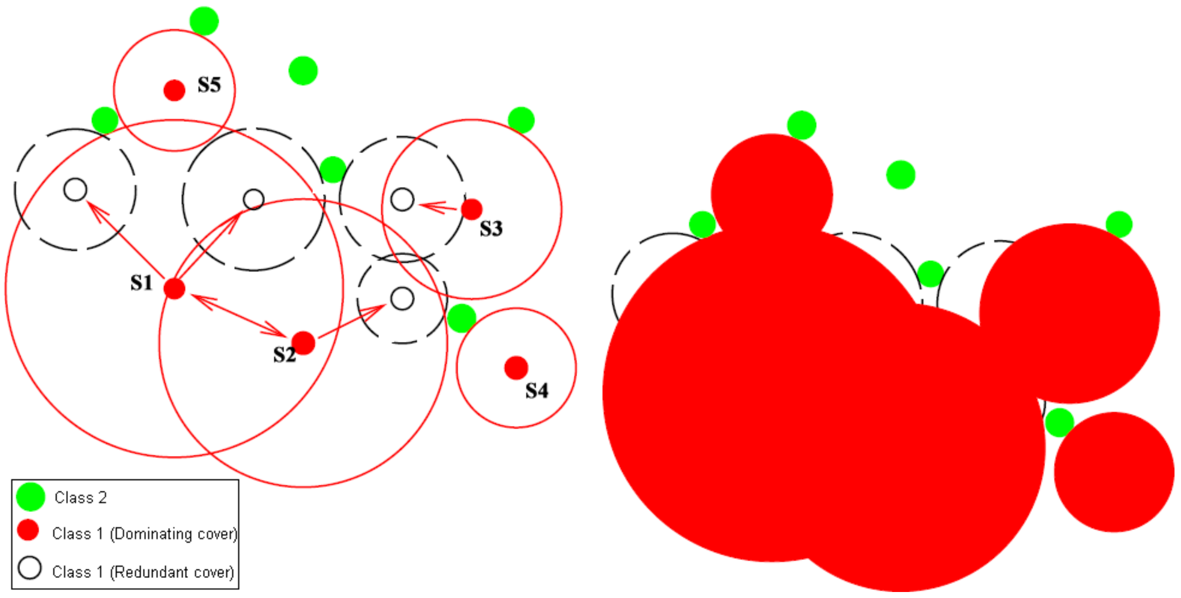


Figure 2.6: A sphere digraph showing the minimal dominating set $D = \{S1, S2\}$ required to cover all the vertices (examples). The domination number $\lambda = 2$.

is possible using a greedy method [29]. A greedy CCCD algorithm (see Algorithm 1) selects a sphere (vertex) from the training set that covers the largest number of examples (vertices). This selection is repeated to include only the examples (vertices) which cover the largest number of uncovered examples not encountered yet.

The greedy CCCD algorithm 1 finds an approximate dominating set which is not unique since it is not specified how to select among equivalent vertices in step 2(a). In general, a random selection between several size spheres is sufficient. In some special cases a more elaborate technique is used [91]. Finding the minimum dominating set is possible for small datasets. However, an algorithm that was proposed in [see 91, algorithm 3.3 in page 134] is unusable in practice.

We can identify the CCP and the CCCD with data reduction methods [18, 99, 141, 142]. Data reduction methods specific purpose is to improve classification and reduce memory load by choosing a specific subset from the training set [17, 142]. Similarly, the CCCD chooses representative or prototypes as covers for the entire class. In the process, the number of training examples stored for classification is reduced. It is possible to build a cover for one class, use a CCCD to find a dominating set for that class, then use the same process for the



(a) An example of a CCCD dominating cover (b) The decision surface of the dominating cover

Figure 2.7: A toy dataset showing a CCCD with a dominating decision surface

Algorithm 1 A Greedy CCCD

```
1: Set $C' = \oslash$, $V = \oslash$.
2: While $C' \neq V$ do:
    • (a) Select a vertex covering the largest number of uncovered vertices
          $v \in \arg\max_{v \in V \setminus \mathcal{D}} |N[v] \setminus C'|$
    • (b) Set $\mathcal{D} = \mathcal{D} \cup \{v\}$ and $C' = C' \cup N[v]$
5: Return the dominating set $\mathcal{D}$
```

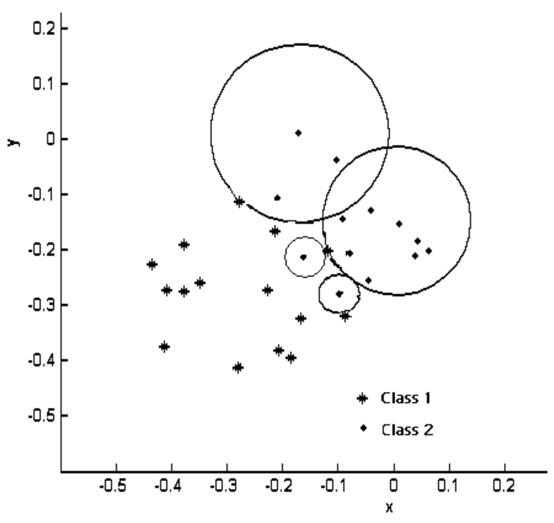
remaining classes. Naturally, this is done at the expense of further running time. Normally, the greedy algorithm requires $O(n^2)$ operations where n is the number of examples in the training set. Marchette [28] showed that for data sizes of many real problems $O(n^2)$ digraph algorithms are not feasible for multiple class problem; this is because we need to calculate the distance matrix of the whole training set for each class. Therefore, further approximation is required at the expense of finding a small dominating set. In addition, a major problem with a pure and proper cover is overfitting [29]. The next section describes two parameters for the CCCD that allow a trade-off between complexity and accuracy [29].

2.1.3.2 (α, β) parameters for the Class Cover

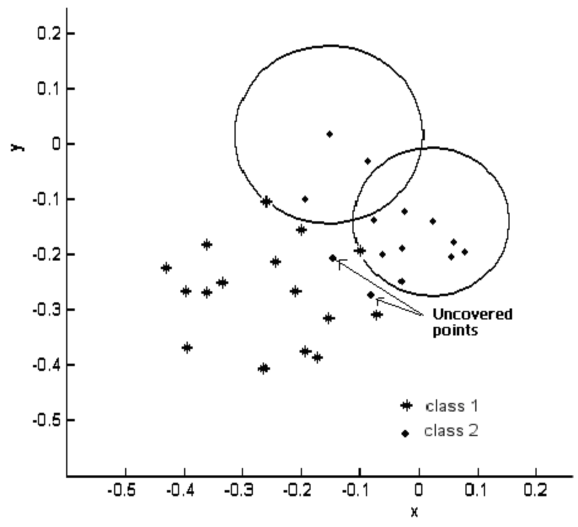
The complexity of the discriminant surface relates to the number of examples chosen to make up the cover (See Figure 2.7). A classifier with the minimum number of spheres should have superior generalisation performance. α and β are two parameters used in the class cover to regularize complex covers [29]. The α parameter is used to prune spheres that are below cardinality threshold α. Removing low cardinality spheres may improve the generalisation performance on noisy data sets. Conversely, β is used to control outliers. In this case, it may be better to allow some spheres to include examples from a different target class. Thus, the β parameter allows the sphere to cover β examples from different class. In other words, β is the parameter that is used to "contaminate" a pure sphere. Increasing α and β results in cover reduction which can lead to better generalisation and reduced data storage (we address the compression issue in chapter 4).

Selecting α values is straightforward and can be tuned using the standard cross validation technique [29]. In [29], a small scale empirical evaluation showed the difficulty of setting β values. It was argued that as β is an outliers filter it would always require small values. Consequently, they proposed a method that chooses a small value of β at the beginning then increments it gradually. In chapter 5, we investigate the β parameter as part of randomizing covers in ensembles and propose an automatic selection process.

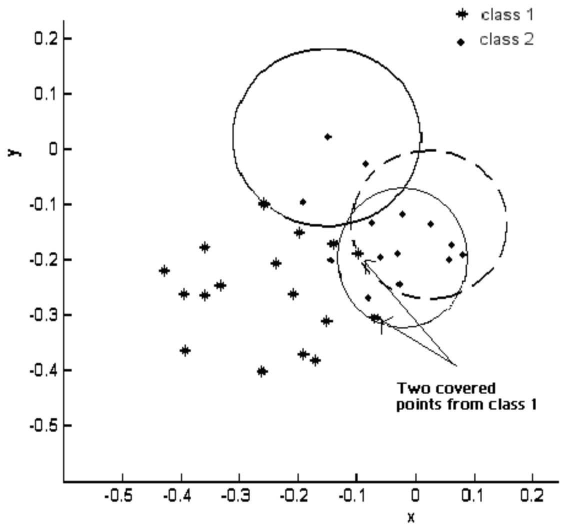
A simple illustration of the role of α and β parameters is shown in figures 2.8. These figures correspond to a single cover of a positive and negative class toy data set. The positive class is represented by points $(\cdot)$ and the negative class is represented by stars $(*)$. In Figure (a), α is equal to 1 and β is equal to 0 which results in a pure and proper cover. In Figure (b), $\alpha = 2$. This will remove spheres that cover only a single examples (singletons).



(a) Pure and proper cover, $\alpha = 1$.



(b) Pure and not proper cover, $\alpha = 2$.



(c) A "contaminated" sphere, $\beta = 2$, the sphere of previous cover is represent in dotted lines.

Figure 2.8: The role of α and β.

In Figure (c), two examples from the negative class are covered by one of the spheres (i.e. “contaminated” sphere).

2.2 The Restricted Coulomb Energy

The Restricted Coulomb Energy (RCE) is an incremental artificial neural network that uses spheres as localized units called receptive regions, whereby the sphere is modelled using as centre the weights and a user defined radius fine tuned during learning [105] (Figure 2.9). The RCE architecture has three layers of nodes. The first layer is the input layer since this is where the inputs are applied. The second layer is called the category representation layer. Training of an RCE, as in any neural network, involves compressing the input space into categories. Consequently, a category is a compressed representation of a group of examples resulting in a generalization of the input space. The number of nodes in the second layer is determined by the training phase. This number increments as training progresses and stabilizes at some value at the end of the training phase. The third layer is the output layer that represents the number of target classes.

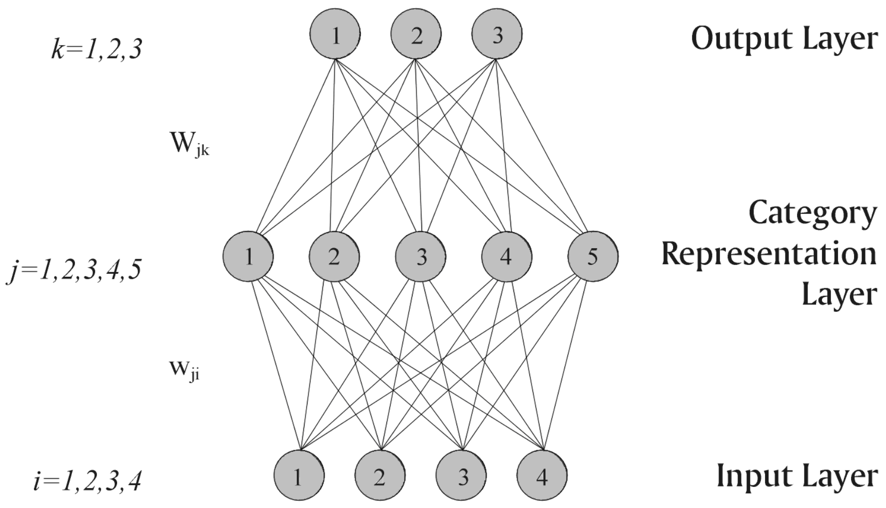


Figure 2.9: An RCE architecture with its three layers. W_{jk} represents the weights of the category representation layer, while w_{ji} represents the weights of the input layer. i is the size of input, j is the size of the spheres found by the network, and k is the number of class label of the dataset

In the training phase, the RCE algorithm requires an initial large fixed radius chosen by the user. New examples are presented to the algorithm in order to modify the radii to ensure that no spheres from other classes contain the example. An example falls inside the sphere (receptive region) if the distance between an example and a category is less than or equal to the radius. A new sphere is created if no sphere covers the newly presented example. We give a general description of the original RCE, disregarding any architectural details, as follows:

- The algorithm keeps a set of spheres with centres from examples it was presented with during learning (called prototypes), so that there is exactly one sphere centred around each prototype
- The radius of a sphere is assigned a large value, called the initial radius.
- The radii of the containing spheres are reduced if a new example presented to the algorithm is contained in some spheres of inappropriate classes, so that none of the spheres contains examples of different class.
- Learning is achieved by presenting examples to the algorithm in order to cover the entire class. These training examples are discarded when they are contained in any existing sphere.
- If an example is not contained in any sphere, a new sphere is created using this example as centre and provided with an initial radius.
- The algorithm does not allow spheres of different class to overlap, whilst this is permissible if they are of the same class.
- The algorithm classifies a new examples using the class of the sphere. Any test example that falls outside a sphere is simply rejected (i.e not classified by the algorithm).

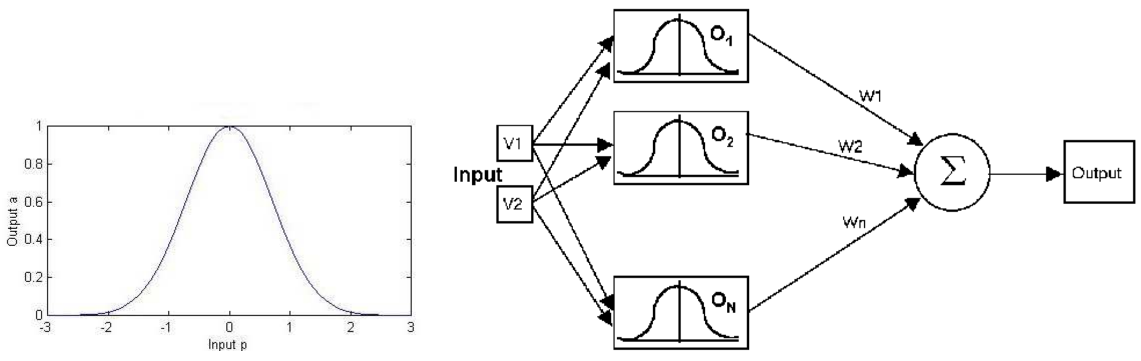
Critics of RCE have found that the learning rule it uses is not well suited for real problems [68]. The reason is that it creates many spheres in the overlapping class region, and therefore does not generalize but merely memorizes the input space. In order to rectify this shortcoming, many variations of RCE have been proposed in the literature [4, 63, and references therein].

2.3 The Radial Basis Function Neural Network

Another popular algorithm that can use spherical shapes is the Radial Basis Function Neural Network (RBFN) [98]. The RBFN uses localized units as spheres, whereby the sphere is modelled using Gaussian functions (Figure 2.10(a)). These Gaussian functions are combined linearly with associated weights to approximate an unknown function. The radial basis transfer function is continuous, in contrast to the Heaviside step function employed in the RCE algorithm.

$$h(\mathbf{x}) = \exp\left(-\frac{(\mathbf{x}-c)^2}{\sigma^2}\right) \tag{2.1}$$

σ, in equation 2.1, is the width of the radial basis function. The centre c of the radial basis function is harder to find. In general, the K-mean clustering algorithm [100], self-organised maps (SMO) [98] and other clustering techniques [87] are used.



(a) Radial Basis transfer function (b) RBF architecture

Figure 2.10: Radial Basis Neural Network.

In order to find the minimum of the loss function, the RBNF algorithms must accomplish the following steps:

1. select a search space.
2. select starting examples in the space.
3. search for the minimum.

An RBFN is completely specified by choosing the following parameters: the number of radial basis functions N, the centres c, the distance function $\mathfrak{d}$, and the weights w_i for each connection between the Gaussian function and the class label ($n > N$) (Figure 2.10(b)). The number N of radial functions is a critical choice and depending on the approach can be made a priori or determined incrementally. RBFN size depends on the number of basis functions: a small number might underfit the data while a large number overfits it. An algorithm that starts with a fixed number of radial basis functions determined a priori is known as static, and an algorithm that is able to modify the number of the basis functions is known as dynamic.

2.4 Ensemble Learning

An ensemble of classifiers (also known as committee of classifiers, committee of learners, mixture of experts, classifier ensembles, multiple classifier system) is a set of classifiers whose individual decisions are combined in some way to classify new examples [32, 95, 103]. In view of the fact that combining the same classifier would be irrelevant, several methods are proposed to generate different hypotheses (or **base classifiers**) [27, 32, 41, 47, 48, 54, 96, 103, 116]. Various empirical results have shown ensembles often improve on the accuracy of these base classifiers [6, 32, 40].

Ensembles that combine classifiers by randomising the training set are called **randomised ensemble methods** [35, 47, 130]. This randomisation is seen as a way to introduce **diversity** in the ensembles. Diversity can be defined as the variability between base classifiers and is a very important concept in ensemble methods [74]. However, the use of diversity is still an open problem in ensemble design [74, 125]. The main reason for this is the weak relation between accuracy and diversity [74]. Nevertheless, many researchers have investigated several **diversity measures** and assessed their link to ensemble accuracy [19, 101, 123].

Diversity can be generated using a range of parameter initialization. If the learning algorithms is deterministic, it can be run several times, each time with a different partition of the training samples. In ensemble methods such randomisation is an essential factor. Dune *et al* [71] outline several possible ways to combine base classifiers. These include:

1. Probabilistically selecting subsets of the training set [Section 2.4.1 and 2.4.2].
2. Introducing artificial examples to the training set to train a base classifier (also known as noise injection or randomness injection) [Section 2.4.3].
3. Randomly selecting subsets from the attribute set [Section 2.4.4].
4. Combining different types of classifiers (also known as heterogeneous ensemble learning).
5. Using different combination schemes.

In order to combine label outputs of several classifiers, we employ some sort of **fusion**. A popular fusion method is the majority vote which combines a number of predictions whereby, as the name suggests, the majority wins [73]. Ties are resolved arbitrarily. Various alternative methods have been proposed and investigated in [75]. Fusion of label outputs is a research area in its own right producing several techniques [77].

We can review a variety of randomised algorithms that combine different classifiers: Noise injection artificially created examples from the training set to train different classifiers in an ensemble; Subset attributes selection employs random selection of attributes whereas each subset is used to build a single classifier for the ensemble; Bagging uses sampling to randomize the input to a classifier; Boosting employs an iterative weighting method to sample or re-weight examples. In the next sections, we describe in more details several of these major algorithms.

2.4.1 Bagging

Bagging is a simple but effective ensemble proposed by Breiman [13]. Bagging is run several times on training samples. On each run, it produces replicates of the original training set by sampling with replacement the same number of examples as the original training set. Some training samples may appear in the produced samples while others may not. Such a training set is called a bootstrap replicate of the original training set, and this technique is called **B**ootstrap **Agg**regat**ing**. Each bootstrap reproduces, on the average, 63.2% of the original training samples [13]. Individual classifiers are then used to classify each example in the test set based on majority vote. Algorithm 2 describes Bagging.

Algorithm 2 Bagging Algorithm

```
1: Input : $D = [(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)]$
2: A learning algorithm $\mathcal{A}$
3: A number of base classifiers (or iterations $T$)
4: Output: The final hypothesis $h_{fin}$
5: for $t = 1$ to $T$ do
6:     $D_t = Bootstrap\_replicate(D)$
7:     $h_t = \mathcal{A}(D_t)$
8: end for
9: $h_{fin}(\mathbf{x}) = argmax_{y \in C} \sum_{t=1}^{T} |h_t(\mathbf{x}) = y|$
```

Given a learning algorithm $\mathcal{A}$ and training set D with m examples, bagging makes D_t

Bootstrap replicates from D. Training a classifier on D_t produces a hypothesis h_t. The final hypothesis $h_{fin}(\mathbf{x})$ is the majority vote of all these predictions.

A variation of Bagging called half&half bagging was also proposed by Brieman [15]. The basic idea is to sample randomly from the training set to form new training sets that comprises of half examples misclassified by previous classifiers, and half of correctly classified examples. The size of the training sets constructed this way is set by the user. An important criteria of half&half bagging is that a misclassified example e would not have been used in any previous sets that trained the classifiers. The example e is labelled using the majority vote of all the classifiers thus far that did not have e in their training set. If the true label of e does not matches the majority vote prediction then it is selected in the misclassified set. This method of labelling and testing examples is termed out-of-bag error. Examples that persist after many draws are shown to be hard to classify. Breiman used the words **hard boundary examples** and showed empirically their significance in classification.

Bagging has shown to work well for unstable learning algorithms [13]. Unstable learning algorithms are those whose output predictions change in response to a small change in the training sample [13].

2.4.2 Boosting

The Boosting method was first proposed by Schapire [117], and was followed by an algorithm called AdaBoost (Adaptive Boosting) [42]. AdaBoost has become the most commonly used algorithm in ensemble methods. Like Halfhalf Bagging, Boosting involves iteratively reweighting the sampling distribution over the training data. The most astonishing characteristic of boosting method is that any weak learning algorithm can be turned into a strong learning algorithm. Boosting was introduced from Probably Approximately Correct (PAC) learning theory (PAC-Boosting). The standard definition of a weak learning algorithms in the PAC-Boosting setting is that it returns a hypothesis h from a fixed set of hypotheses H that is slightly better than random guessing on any training set. Just like bagging, AdaBoost also manipulates the training examples to generate diverse hypotheses. Algorithm 3 show the steps taken by a generic AdaBoost.

Adaboost use a weighting scheme on each example of the training set. A non-negative weighting $\mathbf{d}^{(t)} = (d_1^{(t)}, \dots, d_m^{(t)})$ is assigned to the data at step t, and a weak learner h_t is constructed based on $\mathbf{d}^{(t)}$ which is the weight vector distributed over the training samples. At each iteration t the weights are updated according to the weighted error incurred by the weak

Algorithm 3 AdaBoost Algorithm

```
1: Input : $D = [(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)]$ where $\mathbf{x}_i \in \mathcal{X}$, $\mathcal{X}$ is the set of all examples $\mathbf{x}$, $y \in C = \{-1, +1\}$
2: Initialize $d_n^{(1)} = 1/m$ for all $n = 1, ..., m$
3: Set $\epsilon = 0$
4: for $t = 1$ to $T$ do
5:     Train classifier $\mathcal{A}$ with respect to the weighted sample set $\{D, \mathbf{d}^{(t)}\}$ and obtain hypothesis $h_t : \mathbf{x} \rightarrow \{-1, +1\}$, i.e. $h_t = \mathcal{A}\{D, \mathbf{d}^{(t)}\}$
6:     Calculate the weighted training error $\epsilon_t$ of $h_t$
           $$\epsilon_t = \sum_{n=1}^{m} d_n^{(t)} I(y_n \neq h_t(\mathbf{x}_n))$$
7:     set:
           $$\varphi_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$
8:     Update weights:
           $$d_n^{(t+1)} = \frac{\{d_n^{(t)} exp(-\varphi_t y_n h_t(\mathbf{x}_n))\}}{Z_t}$$
       where $Z_t$ is a normalized factor chosen so that $\sum_{n=1}^{m} d_n^{(t+1)} = 1$.
9:     Break (i.e. do not use the actual classifier) if $\epsilon = 0$ or $\epsilon \geq 1/2$ and set $T = t - 1$.
10: end for
11: Output: $H_T(\mathbf{x}) = \sum_{t=1}^{T} \frac{\varphi_t}{\sum_{r=1}^{T} \varphi_r} h_t(\mathbf{x})$
```

learner $\mathcal{A}$ in the last iteration. The base learner is then applied to produce a classifier h_t. The error rate of this classifier on the training samples is computed

$$\epsilon_t = \sum_{n=1}^{m} d_n^{(t)} I(y_n \neq h_t(\mathbf{x}_n)) \tag{2.2}$$

and is used to adjust the probability distribution on the training samples using the hypothesis weight φ_t,

$$\varphi_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}. \tag{2.3}$$

AdaBoost seeks to minimise the loss function:

$$G(\varphi) = \sum_{n=1}^{m} exp\{-y_n(\varphi h_t(\mathbf{x}_n) + H_{t-1}(\mathbf{x}_n))\}, \tag{2.4}$$

where H_{t-1} is the combined hypothesis of the previous iteration given by

$$H_{t-1}(\mathbf{x}_n) = \sum_{r=1}^{t-1} \varphi_r h_r(\mathbf{x}_n). \tag{2.5}$$

In order to analyse AdaBoost, Shapire et al [117] developed a theoretical tool using the margin theory. In large margin classifiers, the main idea is to enlarge the margin of a linear classifier to obtain good performance [124]. For this reason, margin theory was developed as the main tool of analysis. It is rather surprising to find a link between large margin classifiers and boosting methods. However, boosting uses the notion of hypothesis-margin as opposed to example-margin used in large margin classifiers. The hypothesis-margin is defined as [24]:

> ... a distance measure on the hypothesis class. Therefore, the margin of an hypothesis with respect to an example is the distance between the hypothesis and the closest hypothesis that assigns alternative label to the given example.

The margin Mrg of an example $\mathbf{x}$ with a target class y is calculated as follows:

$$Mrg(\mathbf{x}, y) = \frac{y \sum_t \varphi_t h_t(\mathbf{x})}{\sum_t \varphi_t}. \tag{2.6}$$

The margin of a training example is a number between -1 and +1 that can be interpreted as a measure of the classifier's confidence on this particular example. The choice of φ_t was the critical topic to find out whether maximizing the margin directly has any influence in reducing the generalization error. Early empirical studies showed that AdaBoost achieves large margins[109]. This issue as to whether AdaBoost is a large margin classifier developed into an intense debate for many years [110, 114]. It was only after using a new analytical tool, capturing in the process AdaBoost's dynamic behaviour, that it was possible to study AdaBoost margin. The unique study in [113] showed that AdaBoost does not automatically maximize the margin but asymptotically reaches the maximum margin. Similarly to large margin classifiers, searching for this maximum margin is the main focus of maximum margin boosting [95, 110]. Advances in boosting methods are an example of a continuing growth in interest in this field making boosting methods an exiting research area to study.

2.4.3 Noise Injection

In Noise Injection (NJ), the goal is to explicitly promote classifier diversity by altering the training data. This method differs from Bagging and Boosting in that instead of sampling examples, NJ creates examples from an existing training set. In the past, noise injection has been a popular choice to reduce overfitting in neural networks [48, 121]. In ensemble methods, Raviv and Intrator [111] used bootstrap sampling with noise injection to train neural network classifiers. Liu et al [84] train neural networks in an ensemble using a correlation penalty term in their error functions. Later, this method is referred to as Negative Correlation Learning [20] which became a popular research area in neural network ensembles.

A recent algorithm, called DECORATE, can use either decision trees or neural networks as base classifiers with noise injection to build ensembles [96]. Like boosting, DECORATE is a sequential learning algorithm generating each base classifier iteratively. In DECORATE, decision tree classifiers are trained on the original training data combined with some artificial data generated from the same distribution. The goal, as stated by the authors, is to create diverse classifiers in the ensemble. This is done by keeping the training accuracy of the ensemble high while encouraging diversity. The construction of the artificial data and their labels follow a simple principle: choose labels as to differ maximally from the current ensemble's predictions. At each iteration a classifier will only be accepted in the ensemble if the overall accuracy is increased. This rejection method is prohibitive for large datasets. Experimental results shows that some improvement is made on several datasets when compared to both bagging and AdaBoost.

2.4.4 Subset Attribute Selection in Ensembles

In contrast to the above methods, an effective approach for generating diverse base classifiers is to use different subsets of attributes [102]. Choosing random subsets of attributes is also called Random Subspace method. The main goal of subset attribute selection is to generate diversity in the ensemble [128], since varying attribute subsets will generate different base classifiers. In general, the task of an ensemble generated using an attribute selection algorithm is to: (1) use classifiers generated in different subspaces, and (2) to integrate the predictions in such way to improve generalization. One of the first approaches combined nearest neighbour classifiers (K-NN) through multiple feature subsets (MFS) [7]. MFS was among the first methods to try answering whether less accurate individual classifiers as a whole could achieve high accuracy [7].

Traditionally, finding a set of attributes is known as attribute selection with the main goal of finding the best attribute subset to use for selected learning algorithms [51, 97]. In ensemble methods, the problem is to find the best set of subsets of attributes to maximise diversity. The Filtered Attribute Subspace based Bagging with Injected Randomness (FASBIR) algorithm uses Information Gain (IG) as the search criterion [146] . FASBIR first measures the IG of each attribute then removes all the attributes with information gain less than some threshold. Experimental evaluation showed that a substantial improvement is made in generalisation performance [146].

Using decision trees and nearest neighbour in the subspaces to build ensemble has been shown to be useful in several papers [16, 58, 59]. For example, the random subspace [58] constructs a decision tree based classifier that maintains highest accuracy on training data and improves on generalization accuracy as it grows in complexity. The ensemble consists of multiple trees constructed in randomly chosen subspaces. The popular Random Forests algorithm [16] builds a tree using a bootstrap replica of the learning sample, and a decision tree without pruning. At each test node the optimal split is derived by searching a random subset of size K of candidate attributes selected without replacement from the candidate attributes. Random forest combines randomization with bootstrap sampling. The Random Subspace method is used in a number of ensemble attribute selection strategies. Tsymbal et al [128] survey describes several of these methods which are open for further explorations. This survey is also an indication of the interest given to such stimulating field of research.

2.5 Bias and Variance Decomposition

In general, we can survey three research directions taken in order to analyse the generalization error in ensemble methods. The first research direction is known as the Bias/Variance decomposition which has a rich history in machine learning research [44, 66, 126]. It is the tool we choose to use throughout this thesis. The second research direction is known as the margin theory which is heavily linked to Boosting methods (See section 2.4.2). The third research direction is known as diversity measure [74]. This area has recently been very popular as it was suggested that "diversity" is the missing link for building strong ensemble methods [78]. Research in diversity measures aims at discovering the best way to quantify differences in the base classifiers (i.e diversity) in order to use it in ensemble design [125]. However, it was shown that using some diversity metrics to directly build ensembles is harder than first thought and Kuncheva calls it the “Elusive Diversity” [76].

In general, the bias/variance theory has been used successfully to analyse the error rates of ensemble methods [6, 14, 131]. The main characteristic of the bias and variance decomposition is its simplicity. It is also an appropriate experimental tool used to study the generalization error of any classifier based on a large family of error functions [53, 65]. In fact, the margin theory and diversity in ensembles have all been linked to the bias/variance decomposition [34, 125]. The bias/variance decomposition essentially consists of decomposing the generalization error into two components: bias and variance. Historically, the bias/variance decomposition was used for regression problems, using squared-error as loss function [46]. A general description of the challenging issues to translate the bias and variance decomposition from regression to classification can be found in [65]. In the last decade, a race has taken place to find the best way to achieve bias/variance decompositions for classification where the 0/1 loss function is used [33, 43, 72, 127]. Each proposed method was shown to hold its assumptions and have its shortcomings. In this section, we discuss in more details bias/variance decomposition using Domingos framework [34]. The bias and variance, as defined by Domingos, is for arbitrary loss function but it holds also for the 0/1 loss function [34].

The classical problem in supervised learning is related to both the hypothesis space and the training sample. Figure 2.11 shows three possible hypotheses with their different outcome on a training sample. The dotted black line allows for few mistakes while the hypothesis represented by the non-dotted (green) line shows consistency. The third hypothesis of figure 2.11 shows a simple structure using a linear function. Intuitively, a classifier that performs well on the training set should perform well on unseen data. Bias/variance decomposition shows that this intuition is wrong. Complex hypotheses such as the consistent one in figure 2.11 will have a large variance on unseen data. This is because it fits the data perfectly which makes it prone to small changes in data and may overfit. In figure 2.11, we see that a linear model is far too simple. Simple hypotheses tend to have large bias and often underfit the data (i.e. exhibit bias). An increase in complexity may help reduce bias. In addition, a strong bias in a learning algorithm will mean it is less likely to overfit as it is less dependent on the training sample. The main issue for these algorithms is that an increase in data will not improve the performance since it will not overcome the bias. Alternatively algorithms that are too dependent on the

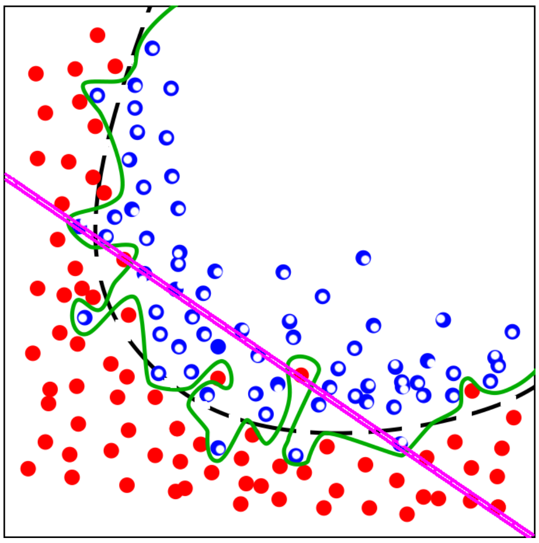


Figure 2.11: Too complex versus simple hypothesis

training sample will have high variance. Overfitting algorithms are known to be unstable and it has been shown that their performance can be greatly improved using ensembles [14, 35].

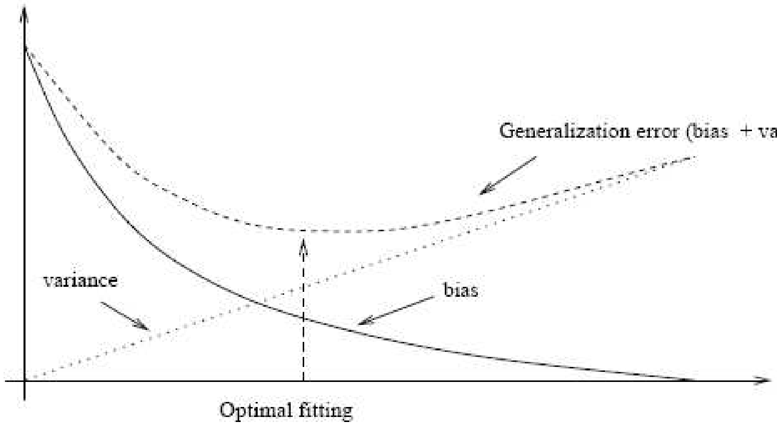


Figure 2.12: Bias/variance trade-off

The aim of any classifier is to find the best fitting hypothesis that decreases the generalization error. This can also be seen as a search to reduce bias and variance. However, this is possible only as a trade-off between bias and variance. This trade-off may be achieved using the parameters of a learning algorithm. Other methods are possible such as restricting the hypotheses space, and even use some prior knowledge to restrict the search space [47]. Figure 2.12 [86] shows the best fitting hypothesis is the one that finds the optimum bias/variance trade-off (optimal fitting). In summary we may say:

$$Generalization\ error = Bias + Variance$$

In this section, we discuss briefly the bias/variance decomposition for the 0/1 loss function using Domingos framework [34].

The bias is attributed to the systematic part of the error, while variance to the stochastic part of the error [34]. It is commonly recognised that:

1. Bias arises when the classifier cannot represent the true function. That is, the classifier underfits the data.

2. Variance arises when the classifier overfits the data.

3. There is often a trade-off between bias and variance.

In practice, the bias and variance are computed by running the algorithm several times on different training sets. To this end, we need to sample repeatedly from a set U in order to make s training datasets $\{D_i\}_{i=1}^{s}$. Each bootstrap D_i is made of l training examples $D_i = \{\mathbf{x}_j, y_j\}_{i=1}^{l}$, where each point is a pair $(\mathbf{x}_j, y_j)$, $y_j \in C$, $\mathbf{x} \in \mathbb{R}^n$, $n \in \mathbb{N}$, and C is the set of class labels. D_i can be considered as a random variable. A learning algorithm $\mathcal{A}$ produces a hypothesis f_{D_i} using a training set D_i such as $f_{D_i} = \mathcal{A}(D_i)$. For each point $\mathbf{x} \in \mathbb{R}^n$ the hypothesis produces a prediction $f_{D_i}(\mathbf{x}) = p$, and $L(y, p)$ represents the 0/1 loss, if $p = y$ then $L(y, p) = 0$, else $L(y, p) = 1$. The goal of our learning algorithm $\mathcal{A}$ consists in minimizing the expected loss EL. Thus, the expected loss at point $\mathbf{x}$ can be written as: $EL(\mathcal{A}, \mathbf{x}) = E_{Di}[E_y[L(y, f_{D_i}(\mathbf{x})]]$, $E_{Di}[.]$ indicates the expected value with respect to the distribution of D_i. $E_y[.]$ is the expectation with respect to y since the randomness in y due to the choice of a particular test point $(\mathbf{x}, y)$.

The two important variables are the optimal prediction p_* and the main prediction (also known as central tendency) p_c. Both p_* and p_c are evaluated using 0/1 loss function and without considering noise (Noise is only considered for theoretical analysis as it is impossible to calculate in practice) [34].

Definition 2.5.1 *(Optimal prediction p_* [34])*
An optimal prediction p_* is the prediction of the optimal classification algorithm (which is the prediction obtained by the Bayes classifier).

In practice we cannot compute this optimal prediction p_* so instead we replace it with y the target value.

Definition 2.5.2 *(Main prediction p_c [34])*
The main prediction p_c for the example $(\mathbf{x}, y)$ is the class most often predicted.

To compute p_c for an example $(\mathbf{x}, y)$ of the test set, we need to get all the $f_{D_i}(\mathbf{x})$ predictions for that example from different hypotheses, and then find the prediction that appears most often, this will be p_c.
The bias $B(\mathbf{x})$ is the loss of the main prediction relative to the optimal prediction p_*.
Bias measures how far the predictions of a learning algorithm, given an example $(\mathbf{x}, y)$, are from the optimal prediction p_*. For the 0/1 loss, the bias is always 0 or 1. Thus, it is said that the learning algorithm $\mathcal{A}$ is biased at point $\mathbf{x}$, if $B(\mathbf{x}) = 1$. The ***bias*** $B(\mathbf{x})$ is:

$$B(\mathbf{x}) = L(p_*, p_c) \tag{2.7}$$

Definition 2.5.3 *(Net Variance $V(\mathbf{x})$ [34])*
The net variance $V(\mathbf{x})$ is the average loss of the predictions relative to the main prediction.

Net variance measures how the choice of the training set affects the predictions of the learning algorithm. In our case, it measures how the predictions of a learning algorithm for a specific example, derived from the D_i different training sets, fluctuate around the most often prediction p_c associated with that example.
The ***net variance*** $V(\mathbf{x})$ is:

$$V(\mathbf{x}) = E_{Di}[L(p_c, f_{D_i}(\mathbf{x}))] \tag{2.8}$$

The **biased variance** V_b and the **unbiased variance** V_u constitute the two components of the net variance. The unbiased variance corresponds to the variance of incorrect predictions for the case where the main prediction is correct ($p_c = p_*$). Thus, unbiased variance captures the extents to which the learner deviates from the correct prediction p_c. In this case, the unbiased variance is added to the error. On the other hand, the biased variance corresponds to the variance of correct predictions for the case where the main prediction is incorrect ($p_c \neq p_*$). Thus, biased variance captures the extents to which the learner deviates from the incorrect prediction p_c. As a consequence, the net variance is the difference of the two: $V = V_u - V_b$. This means that variance hurts on unbiased examples while it helps on biased examples.

The b/v decomposition is:

$$EL(\mathcal{A}, \mathbf{x}) = c_1 N(\mathbf{x}) + B(\mathbf{x}) + c_2 V(\mathbf{x})$$

The noise part $c_1 N(\mathbf{x})$ is disregarded simplifying the decomposition to

$$EL(\mathcal{A}, \mathbf{x}) = B(\mathbf{x}) + c_2 V(\mathbf{x})$$

c_2 is $+1$ if $B(\mathbf{x}) = 0$ and -1 if $B(\mathbf{x})) = 1$.

Thus, the average loss $E_{\mathbf{x}}[EL(\mathcal{A}, \mathbf{x})]$ for a learning algorithm $\mathcal{A}$ on all the examples is calculated using the average bias, variance (unbiased, biased and net variance), averaged over the entire set of the examples of the test set is:

$$\begin{aligned} E_{\mathbf{x}}[EL(\mathcal{A}, \mathbf{x})] &= E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[V_u(\mathbf{x})] - E_{\mathbf{x}}[V_b(\mathbf{x})] \\ &= E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] \end{aligned} \tag{2.9}$$

To give a simple interpretation, we use a similar illustration presented in [135]. Let $(\mathbf{x}, y)$, be an example where $y \in C = \{a, b, c\}$ is the target value of an example $\mathbf{x}$.

Table 2.1: Table showing an example of BV calculation

	Case1	Case2	Case3
Correct class	a	b	c
Prediction 1	a	a	a
Prediction 2	a	a	b
Prediction 3	a	a	c
Prediction 4	a	a	c
Prediction 5	b	a	c
Prediction 6	b	a	c
Prediction 7	b	b	c
Prediction 8	c	b	c
Prediction 9	c	b	c
Prediction 10	c	b	c
Main prediction	a	a	c
Bias	0	1	0
Variance	0.6	0.4	0.2
Error	0.6	0.6	0.2

Let say that an algorithm is run 10 times on different training sets. For each example $(\mathbf{x}, y)$, we get a prediction, for a total of 10 predictions as shown in table 2.1. The main prediction

for an example $(\mathbf{x}, y)$ is the class most often predicted. For the 0/1 loss, the bias is always 0 or 1. The contribution of bias to error depends on the loss of the main prediction relative to the optimal prediction. The contribution of variance to error depends on the average loss of the predictions relative to the main prediction. Thus, the error in Domingos bias/variance decomposition is:
Case 1 : $E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] = 0 + ((1 - 0) * 0.6) = 0.6$
Case 2 : $E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] = 1 + ((1 - 2) * 0.4) = 0.6$
Case 3 : $E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] = 0 + ((1 - 0) * 0.2) = 0.2$

In the second case, the error comes from both bias and variance, whereas in the two other cases, the error comes from variance only. As stated above, the interesting point about Domingos decomposition is that reducing unbiased variance in case 1 will help reduce variance. Hence, the overall error is reduced. In the other hand, reducing the biased variance of case 3 will increase the overall error. It becomes clear that in order to reduce the overall error, it is required that both bias $(B(\mathbf{x}))$ and unbiased variance $(V_u(\mathbf{x}))$ are reduced.

Chapter 3

The Randomized Sphere Cover Classifier (αRSC)

3.1 Introduction

In this chapter, we propose a new randomised sphere cover classifier (αRSC), based on both the CCP and CCCD characteristics, described in Section 2.1.3, which is fast in both learning and classification. The objective of our endeavour is to investigate the proposed classifier in order to use it in ensemble design. To this end, we empirically test the performance of αRSC and compare it against alternative classifiers. We show the αRSC performs comparably with, or better, than five other classifiers on 24 data sets. Furthermore, we use bias/variance decomposition to analyse the generalisation error of αRSC. Finally, a number of issues are discussed and possible solutions to tackle in future.

In Chapter 2, we described a family of classifiers called Instance-Based Learning (IBL). Even though these classifiers are well established in the machine learning literature, employing them as base classifiers for ensembles has not been straightforward [119]. Instance based learning techniques operate by keeping a typical sample of the training data then classifying new examples based on their similarity to the retained sample. Instance based learning algorithms are defined by three characteristics: a similarity function that specifies the closeness of two examples, a selection function that selects the samples to be kept by the algorithm, and a classification function that decides on the class of unseen examples. The simplest and most popular IBL algorithm is the nearest neighbour (NN) algorithm which retains the entire training set. Although surprisingly effective, one well documented problem with NN classifier is that classifying a new example requires a distance calculation for each example in the training set. Data reduction algorithms have been studied in great depth [10, 11, 17, 70, 142]. In general, these algorithms search the training data for a subset of cases and/or attributes with which to classify new examples with the objective of achieving the maximum compression with the minimum reduction in accuracy. In this chapter, we propose a simple and fast randomised data reduction algorithm (αRSC) that creates spheres around a subset of examples, then bases classification on distance to spheres rather than examples.

The reason for designing the αRSC algorithm was to develop an instance based classifier to use in ensembles. Ensemble performance depends on many factors present in the base classifier. Notably, the stability of the classifier [12, 37], and on their general geometrical properties [60, 62]. In addition, various empirical results showed that the choice of a base classifier in ensembles can have a significant effect in accuracy [3, 8, 120, 129]. Hence our design criteria were that the base classifier should be randomised (to allow for diversity) and fast (to mitigate against the inevitable overhead of ensembles) and comprehensible (to help produce meaningful interpretations from the models produced).

The research presented in this chapter was published in the 11th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2010) and Int. J. of Data Mining, Modelling and Management [144, 145].

3.2 A Randomized Sphere Cover Classifier (αRSC)

The CCCD described in Section 2.1.3 uses a greedy method which requires $O[(n^2 + nm)d + \gamma(n+m)]$ [28], where γ is the size of the resulting dominating set, n is the number of examples from class C_1, m represents the number of examples from class C_2 and d is the number of attributes of an example $\mathbf{x}$. The dominant term $[O(n^2+nm)d]$ comes from the distance matrix calculation and the second term $\gamma(n + m)$ comes from the selection of the dominating set. A greedy CCCD requires two searches of a training set. The first search is to determine the radius of each sphere that make up the entire training data for the chosen class. The second search is to find examples that are covered by each sphere. This is clearly an issue for complex and large data sets. The search for an approximation has been the main focus of its inventors [22].

A fast randomized algorithm that selects an approximate dominating set for the CCCD which does not require the explicit calculation of graphs is described in [28]. The randomized algorithm builds a sphere around centres randomly selected from the training set. The randomized algorithm has an $O[\gamma'(n+m)d]$, where γ' is the size of the resulting dominating set of the randomized algorithm and $(\gamma' \geq \gamma)$ (i.e. γ' is far bigger in size in comparison to γ). The randomized algorithm is faster than a greedy algorithm but may return a larger dominating set than the greedy approach, especially for data sets that have high class overlap [28]. It has been shown that the computational complexity would be the same for both algorithms, and it may be better to use the greedy approach, if the data set is small [28]. We are particularly interested in the randomised algorithm for three main reasons:

1. The algorithm is fast to cover the training set which drives the training error to zero. However, the speed to cover the entire training set depends on the attribute distributions for each class, i.e. dense and well separated data sets will be much easier to cover [28].
2. We wish to exploit this randomization to build covers for ensemble methods since one objective is the design of ensemble methods is to diversify the members (more details in Chapter 5).
3. It would be fast to build ensembles using a relatively large number of these randomised classifiers.

Consequently, we shift our focus from the problem of searching and selecting the best set of spheres for a cover, to a random search and selection problem for ensemble design.

The fast randomized algorithm described here is not a classifier. To make the fast randomized cover a suitable classifier, two important factors are taken into consideration:

1. The choice of regularization parameters.
2. The classification rules.

The αRSC algorithm has a single integer parameter, α, that specifies the minimum size for any sphere. Informally, for any given α, αRSC works as follows.

1. Repeat until all data are covered
   (a) Randomly select a data example and add it to the set of covered cases.

(b) Create a new sphere centred at this example.

(c) Find the closest case in the training set of a different class to the one selected as a centre.

(d) Set the radius of the sphere to be the distance to this case.

(e) Find all cases in the training set within the radius of this sphere.

(f) If the number of cases in the sphere is greater than α, add all cases in the sphere to the set of covered cases and save the sphere details (centre, class and radius).

A more formal algorithmic description is given in Algorithm 4. For all our experiments we use the Euclidean distance metric, although the algorithm can work with any distance function.

Algorithm 4 A Randomized Sphere Cover Classifier (αRSC)

```
1: Input: Examples $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, distance function $\mathfrak{d}(\mathbf{x}_i, \mathbf{x}_j)$ parameter $\alpha$.
2: Output: Set of spheres $B$
3: Let $A = \oslash$ ($A$ being the set of covered cases)
4: **while** $D \neq A$ **do**
5:    Select a random element $(\mathbf{x}_i, y_i) \in D - A$
6:    Copy $(\mathbf{x}_i, y_i)$ to $A$
7:    Find $\min_{(\mathbf{x}_j, y_j) \in D} \mathfrak{d}(\mathbf{x}_i, x_j)$ such that $y_i \neq y_j$
8:    Let $r_i = \mathfrak{d}(\mathbf{x}_i, x_j)$
9:    Create a $B_i$ with a centre $\mathbf{c}_i = \mathbf{x}_i$, radius $r_i$
      and target class $y_i$
10:   Find all the cases in $B_i$ and store in temporary set $T$
11:   **if** $|T| \geq \alpha$ **then**
12:      $A = A \bigcup T$
13:      Store the sphere $B_i$ in $B$
14:   **end if**
15: **end while**
```

Algorithm 4 randomly selects an example from the training set without replacement. That is, once an example is selected as a potential centre for a sphere, it is not reconsidered. The radius of a sphere is defined as the distance to the closest example of different target class which is called the border example. Each sphere is assigned one border example. The classification rule is described as follow:

1. **Rule 1.** A test example that is covered by a sphere takes the target class of the sphere. If there is more than one sphere of different target classes covering the test example, the classifier takes the target class of the sphere with the closest centre.

2. **Rule 2.** In the case where a test example is not covered by a sphere, the classifier selects the closest spherical edge to the test example and uses that sphere class label as the prediction.

A case covered by Rule 2 will generally be an outlier or at the boundary of the class distribution. Therefore, it may be preferable not to have spheres over covering areas where such cases may occur. These areas are either close to the decision boundary where there is high overlap between classes (an illustration is given in Figure 3.1 (a)), or in areas where noisy cases are within dense areas of examples of different target class. The αRSC method of compressing through sphere covering and smoothing via boundary setting provides a robust simple classifier that is competitive with other commonly used classifiers. In general:

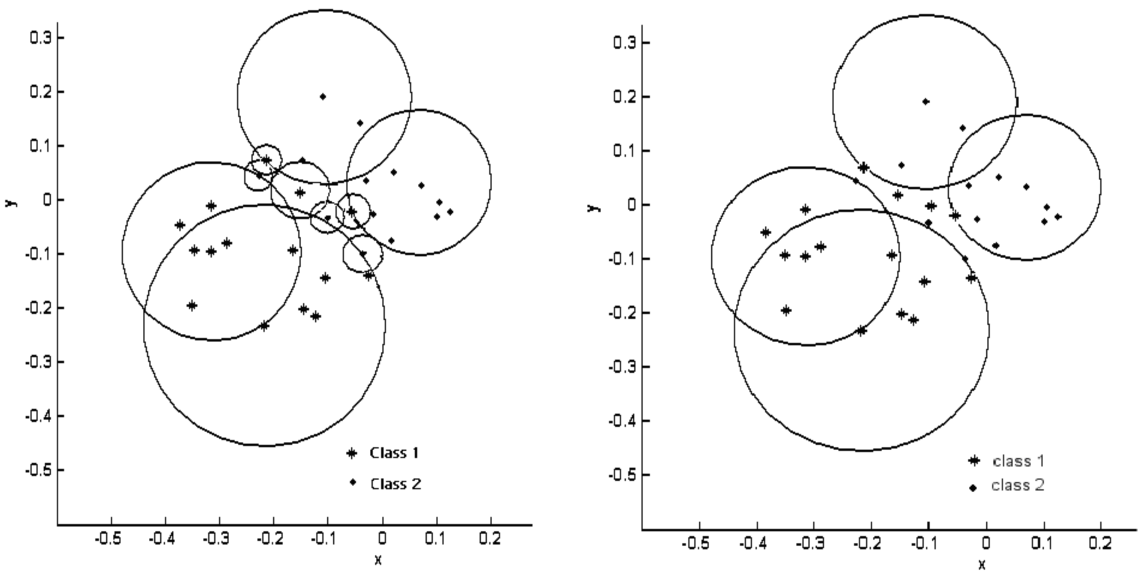
- Spheres that have big radii and cover large number of examples have centres selected from examples far away from the decision boundary.

- Spheres that have big radii and cover very small number of examples may be outliers.

- Small spheres are either close to the decision boundary or they are noise found in dense area of a different target class.

We show the reason this characterisation of spheres using the proposed classifier is important in our ensemble method described in Chapter 5.

We described, in the previous Section 2.1.3, that pruning spheres is the regularization parameter of choice. A further illustration is shown in Figure 3.1 (b). Removing singleton spheres from a cover may result in good surface separation. The authors that proposed the second regularisation parameter β [29] (see Section 2.1.3.2) showed no clear evidence for its efficiency. Although it is clear that removing some outliers will certainly reduce the number of spheres in a cover, it is also clear that no rule of thumb exists to choose values for β. In practice, this makes it rather difficult to use β, since β should be a "local"parameter that may require a different value for each sphere. Therefore, at this point we avoid using the β parameter. However, It may be helpful if we can develop a strategy to automatically select values for β. This strategy will be investigated in Chapter 5 as part of a new ensemble method



(a) Pure and proper cover showing complex decision boundaries

(b) The same cover with $\alpha = 1$ resulting in simpler decision boundaries

Figure 3.1: An example of complex versus simple cover (both being target class)

(we also test, in the same chapter, our hypothesis that automating β may generate more cover diversity).

Marchette [91] proposed a variety of ways to make classifiers from the class cover problem. As far as we know, very limited experimental evaluation was carried out to study their performance as most of them are impractical for large and complex data sets. To the best of our knowledge, the proposed randomized sphere cover classifier has never been investigated nor has it been implemented as used in this thesis. In addition, no bias/variance decomposition has ever been employed to study any class (sphere) cover classifier.

3.3 Experimental Evaluation of the αRSC

In this Section, we investigate the accuracy and generalisation performance of αRSC on 24 benchmark datasets of table 1.1 in Section 1.4. Section 3.3.2 explores the simple version of the proposed classifier with fixed α and compares the accuracy results against those of the nearest neighbour classifier. In Section 3.3.4.1, we look at the learning curve of αRSC in relation to the pruning parameter α. Section 3.3.4.2 investigates the bias/variance trade-off of αRSC, and the role of bias and variance in the reduction of the generalisation error. Section 3.3.5 compares the accuracy results of αRSC with variable α against those of other well known classifiers.

3.3.1 Experimental Setup

For the experiments in Section 3.3.2 we use 2/3 of the data set for training and tested. We use the training for model selection and report the best classification on test set alone. For the experiments in Section 3.3.4 we used a stratified ten-fold cross validation (CV). In Section 3.3.5 we run the algorithms using 30 random splits of the training and testing sets. For comparison purposes we used K-NN, the Non Nested Generalized Exemplar (NNge) [92], C4.5, Naive Bayes and NBTree. K-NN and NNge are the most relevant for comparison purposes, the other three are included for completeness. WEKA [143] implementations are used for the standard classifiers, except our implementation for αRSC.

The accuracy results provided in Section 3.3.5 are based on an independent test set drawn randomly from the data set. We use 2/3 of the data set for training and tested the classifiers on the same remaining data. However, given the randomisation nature of αRSC, we choose to use the average of 30 runs in order to make a fair comparison. Tuning the parameters for both α and K, in Section 3.3.5, is based on 10 CV using the training set alone and the average of 15 experiments. NNge was trained based on the best parameters suggested by its authors. The decision tree (J48) is unpruned and it was trained using the default parameters in WEKA [108], which are also suggested by its inventor. Naive bayes has no parameters.

We are primarily interested in the relative performance of the classifiers over the range of data sets. In order to compare the algorithms over all datasets, we use Friedman ranks sum test [67]. This test ranks the classifiers over each dataset (with the best performing algorithm getting the Rank of 1, the second best rank 2, etc.). Let r_{ij} be the rank of the j^{th} of k algorithms on the i^{th} of N data sets. The average rank of classifier, $R_j = \frac{1}{N}\sum_i r_{ij}$ gives a non-parametric summary of the relative performance over all the data sets, and it has been shown that the ranking themselves provide a fair comparison of the algorithms [67].

The Friedman test checks whether the measured average ranks are significantly different from the expected ranks under the null hypothesis. The null-hypothesis states that all algorithms are equivalent and their ranks should be equal on average. If the null hypothesis is rejected we proceed to post-hoc pairewise test [67] such as the Nemenyi test [104]. The Friedman statistic is distributed according to the Chi square with $k-1$ degrees of freedom, when N and k are big enough ($N > 10$ and $k > 5$). For smaller values an exact critical value can be computed.

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_j R^2 j - \frac{(k(k+1)^2}{4} \right] \tag{3.1}$$

The Friedman test checks whether the measured average ranks are significantly different from the mean rank. The hypotheses are:

- Ho: There is no difference in mean ranks for repeated measures.
- Ha: A difference exists in the mean ranks for repeated measures.

3.3.2 Experiment 1: Compare αRSC and Greedy CCCD

Our first experiment compares the accuracies produced by the randomised sphere cover classifier to those produced by the greedy CCCD. Instead of using the randomised method to select the set of spheres, we use the greedy method and keep the same parameter and classification rules of algorithm 4. We used the binary class datasets of table 1.1. The results in table 3.1 indicate that no advantage is gained using the greedy CCCD. In fact, αRSC performs slightly better. These experiments also shows that the greedy CCCD has no advantage over αRSC on data compression rate as shown in table 3.1. Data reduction is calculated as follow:

$$\text{Comp } (\%) = 100 \times \frac{|B|}{|S|} \tag{3.2}$$

In some cases data reduction in the greedy method is better than αRSC but we expected a systematic win of the greedy CCCD on all the dataset since the greedy method retains the smallest subset. The main advantage is the randomisation can reduce the training time since no computation is required for the graphs to find the dominating set. It is also interesting to note that α on both classifiers varies but with not large difference.

3.3.3 Experiment 2: Evaluation of the Basic αRSC

Our second experiment demonstrates that even by using the most basic form of RSC with $\alpha = 1$ we can massively compress the data without significantly reducing the accuracy compared to an instance based learner using the full data set. We ran a tenfold cross validation on each of our 24 data sets. Table 3.2 shows the accuracy on the test data. Table 3.2 also shows the average compression rate achieved for each data set.

The average compression rate for unpruned RSC is 75%. These experiments clearly show that by using the simplest form of αRSC we can discard a large proportion of the data whilst maintaining the same level of accuracy. In order to verify the compression rate and the accuracy of the pruned RSC (αRSC), we compared it against Drop3, IB3 and Explore. Drop3 was shown to be the best reduction algorithm in terms of reduction and accuracy in comparison to 14 other data reduction algorithms [142]. IB3 is an algorithm which was proposed to rectify shortcomings of the famous IB1 and IB2 [142]. Explore was shown to produce very good reduction without too much deterioration in accuracy [142]. The results produced in Table 3.3 shows the best accuracy produced by K-NN and αRSC trained over a range of parameter values while the reduction algorithms use internal tuning to produce the best results.

The results shown in Table 3.3 demonstrate once more the good performance of αRSC in comparison of state of the art data reduction algorithms. In addition, the average accuracy is comparable to that of K-NN.

Table 3.1: Classification accuracy of αRSC and Greedy CCCD using the same train/test splits. Only binary class datasets are used and the classifiers are run once. Data reduction (Comp %) is also shown

Dataset	αRSC	Comp %	α	Greedy CCCD	Comp %	α
banana	89.08	96.77	7	89.02	97.48	7
cancer	78.49	96.20	7	77.42	96.20	7
clouds	88.48	94.27	4	88.00	92.44	5
concentric	98.44	97.12	2	98.20	98.74	0
diabetes	76.17	92.19	5	77.34	87.30	5
german	76.05	81.53	2	75.15	88.14	3
glass2	97.22	90.85	2	95.83	95.77	3
haberman	74.51	88.56	3	75.49	88.56	2
heart	86.67	92.78	8	87.78	94.44	9
ionosphere	91.45	67.09	0	94.02	74.36	0
liver	63.48	89.57	3	64.35	87.83	3
phoneme	88.68	74.96	0	88.68	82.15	0
ringnorm	96.35	80.84	1	96.88	85.16	1
segment	96.75	91.82	2	96.62	93.83	1
sonar	78.57	73.19	1	78.57	77.54	1
thyroid	97.22	83.22	0	95.83	58.74	0
twonorm	96.68	99.94	600	97.28	98.58	300
wdbc	96.32	92.61	6	95.26	91.56	2
wins	98.28	94.85	7	98.28	92.70	8

Table 3.2: 10 fold Cross-Validation classification accuracy (in%) and standard deviation over the folds. The final column gives the rounded average compression rate (Comp) for unpruned RSC (in%). 1-NN stands for the Nearst Neighbour. Comp for the compression rate.

Data Set	1-NN	unpruned RSC	Comp %	Data Set	1-NN	unpruned RSC	Comp %
vehicle	69.61 (4.62)	68.13 (4.75)	50	glass6	70.30 (8.96)	69.00 (9.49)	52
segment	97.14 (1.07)	96.10 (1.21)	89	cancer	67.65 (7.80)	68.08 (7.76)	52
abalone	50.13 (2.25)	49.46 (2.02)	32	breastw	95.67 (2.48)	95.36 (2.42)	90
waveform	85.88 (1.57)	85.41 (1.55)	73	concentric	98.54 (0.79)	98.21 (0.82)	97
ringnorm	72.59 (0.53)	95.16 (0.49)	63	clouds	84.64 (1.68)	84.75 (1.48)	76
magic	80.16 (0.32)	79.95 (0.35)	68	wdbc	94.01 (2.95)	95.38 (2.65)	88
pendigits	98.95 (0.10)	97.72 (0.25)	93	thyroid	96.80 (3.33)	95.40 (4.44)	88
vowel	98.90 (1.05)	95.70 (2.34)	77	german	70.70 (4.34)	70.30 (3.86)	52
twonorm	94.51 (0.29)	93.78 (0.34)	83	diabetes	70.62 (4.67)	68.87 (5.02)	51
glass2	94.25 (4.72)	93.86 (5.67)	87	ionosphere	87.10 (5.12)	92.80 (3.75)	69
ecoli	80.66 (6.16)	81.75 (6.26)	66	heart	75.78 (7.34)	75.26 (8.98)	60
haberman	65.77 (6.92)	68.58 (7.38)	53	sonar	86.23 (7.41)	82.80 (8.48)	61

Table 3.3: 10 fold Cross-Validation classification accuracy (in%). The Comp columns give the average compression rate (Comp) for αRSC (in %).

Dataset	K-NN	αRSC	Comp %	IB3	Comp %	Drop3	Comp %	Explore	Comp %
sonar	87.02	81.22	80.28	76.57	83.07	80.38	79.49	77.45	98.88
glass2	94.37	94.76	90.50	93.42	88.42	94.83	92.16	92.99	98.96
glass6	72.42	70.44	75.65	64.03	67.97	67.34	76.12	67.81	96.57
tyroid	95.76	94.26	95.32	91.65	91.27	93.94	89.20	92.90	98.50
heart	83.33	82.81	99.09	58.89	93.21	81.85	80.16	83.33	99.18
haberman	73.56	74.89	91.41	26.48	98.98	73.88	91.65	73.19	99.56
cancer	74.11	74.40	93.09	39.48	95.92	74.50	90.69	68.36	99.48
ecoli	84.84	85.09	81.29	81.86	70.08	84.24	81.25	83.03	98.08
iono	85.75	93.40	78.74	85.49	86.17	86.03	92.97	79.77	99.02
wdbc	96.31	96.26	92.91	93.50	90.57	95.60	89.87	95.78	99.61
wins	96.57	97.03	95.97	96.28	93.96	96.28	95.55	96.43	99.68
diabetes	73.70	74.63	82.29	70.30	90.26	75.66	82.15	74.48	99.71
vehicle	71.26	66.23	83.84	65.48	72.60	68.79	75.85	47.87	99.29
vowel	99.09	93.16	79.01	93.94	79.28	94.65	70.38	71.01	93.29
german	75.30	73.87	89.30	70.50	90.19	73.60	83.60	69.40	99.78
conc	98.68	98.64	98.33	97.00	93.00	98.28	91.11	63.16	99.96
image	97.71	96.20	89.96	94.42	93.11	95.76	91.39	87.75	99.59
abalone	53.77	54.44	92.16	53.05	80.37	54.78	82.86	53.00	99.92
clouds	88.52	88.81	95.26	87.26	95.37	88.10	93.10	77.94	99.96
waveform	88.80	89.56	99.44	86.26	96.83	89.28	87.20	85.36	99.96
ringnorm	72.45	95.60	81.37	86.18	85.58	91.69	92.88	86.19	99.46
twonorm	97.24	96.59	98.98	95.72	96.82	96.77	90.69	95.92	99.95
pendigitis	99.07	97.83	94.24	97.39	94.80	97.85	94.13	95.27	98.89
magic	83.53	83.12	89.48	80.10	95.44	83.70	89.06	76.50	79.88
average	85.13	85.55	89.50	78.55	88.47	84.91	86.81	78.95	98.22

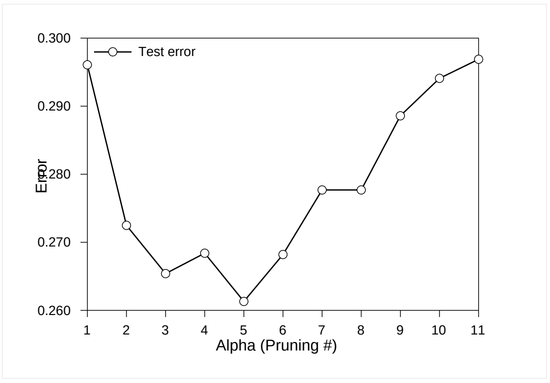
3.3.4 Experiment 3: Model Selection for αRSC

3.3.4.1 Learning Curves of αRSC

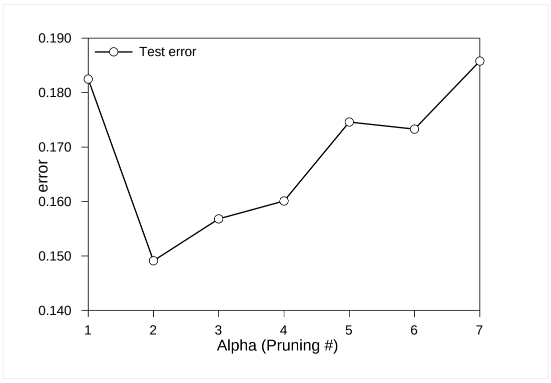
The main factor discussed in this Section is the different methods for choosing values for α. Figure 3.2 shows the learning curves of αRSC in relation to the pruning parameters α (The 10 CV classification error averaged across 100 runs). The curves in Figure 3.2 show that small pruning values may give significant difference in error, as shown for curves (a) and (b) of Figure 3.2. We also notice for the same curves a sharp increase in error for $\alpha > 5$ and $\alpha > 2$. The exception is shown for curve (c) of Figure 3.2, since Heart dataset is small, it required large α value which may be the source of high class overlap (noise). For large data sets, selecting α values share the same difficulty. In some cases larger values may be needed such as the one shown for curve 3.2(d). The choice of α is clearly data dependent. In addition, we notice that further pruning in some cases stabilizes up to a certain value, and then the increase in error is sharp. By looking at these curves, it becomes evident that pruning depends on the geometrical properties of the data sets. The geometrical properties of dataset might be analysed using the geometrical complexity analysis [61] which gives estimates of the degree of noise, and it also gives an analysis of the structure's complexity in datasets. Most importantly, if the class distribution is unbalanced it may require different α values for each class.

Choosing the right values for the pruning parameter should therefore consider the size and noise level of a data set. In general, pruning a large training set may require large α values (as long as the data set is not too noisy), and small data sets should require small α values. In this chapter, we employed 10 fold cross validation for selecting a value for α for each dataset. We first split the dataset into an independent training and test sets. The cross validation is employed only on the training set. Finally, the chosen α value is the one that gave the best 10 CV classification result on the training set.

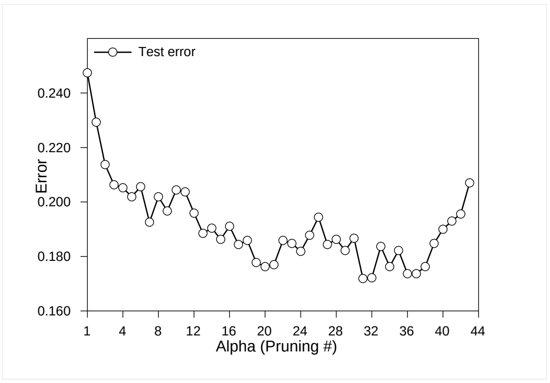
We also notice in Figure 3.2 a common U-like curve. This trend explains the overfitting issue we explained earlier. Unpruned covers result in complex decision surfaces. As spheres are removed from the cover better generalisation is achieved. Pruning works better for data sets that are complex in nature. In classification, complex datasets have noisy examples and could have high class overlap. Conversely, the generalisation performance deteriorates if it is severely pruned. We will investigate some of these cases from a bias and variance perspective to give us a better understanding of the overfitting and underfitting issues of αRSC.



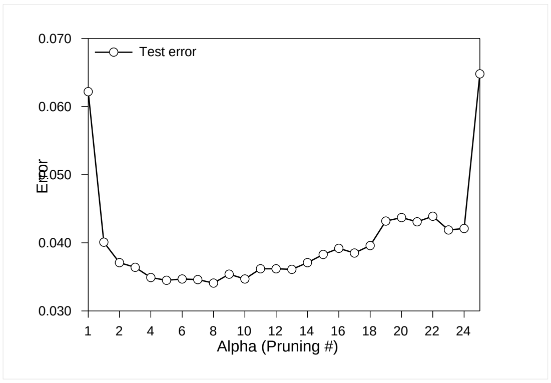
(a) Learning curve of αRSC on the German data set



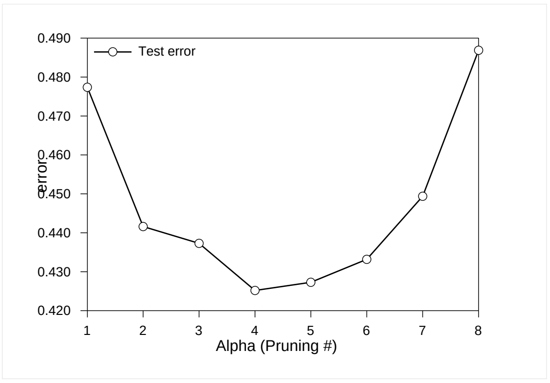
(b) Learning curve of αRSC on the Glass6 data set



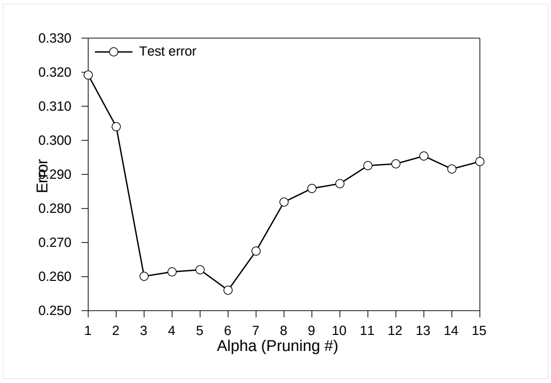
(c) Learning curve of αRSC on the Heart data set



(d) Learning curve of αRSC on the Waveform data set



(e) Learning curve of αRSC on the Yeast data set



(f) Learning curve of αRSC on the Cancer data set

Figure 3.2: Learning curves evolution of αRSC classifier in relation to α

3.3.4.2 The Bias and Variance Decomposition of the Error for αRSC

The aim of this section is to study the generalisation error of αRSC using bias/variance decomposition. For these experiments, we employ ten different data sets, four synthetic and the remaining data sets are taken from the UCI repository. Clouds and Concentric are synthetic two dimensional two-class data set. Both of these data sets are taken from Elena project[1]. Table 3.4 summarizes the main features of the data sets used in the experiments.

In bias/variance decomposition, small training set and large test sets are used to perform the evaluation of bias and variance. For both synthetic and real data sets we used bootstrapping to replicate the data. In both cases we compute the main prediction, bias, unbiased and biased variance, and net-variance (as explained in Section 2.5). We followed a similar experimental framework found in [131]. The data is divided into a training T_r and a test T_s sets. The ratio of the training and test sets are shown in table 3.4. The training bootstrap samples are much smaller than $|T_s|$. That is, 200 data sets are made from T_r, each one consisting of 200 examples uniformly drawn with replacement from T_r. Except for two data sets Twonorm and Waveform, we used 300 training data sets each made of 300 examples.

Table 3.4: data sets used for the bias/variance decomposition of the error. (# of attr.) stands for the attribute size. (# of class.) stands for the number of classes. (# of T_r.) stands for the number of training sets (bootstraps). (# of examples.) stands for the number of examples used for each training set (bootstrap). (Test set ratio.) stands for the proportion of the test set in relation to the whole set

Data set	# of attr.	# of class.	# of T_r sets.	# of examples.	Test set ratio.
Twonorm	20	2	300	300	1/2
Ringnorm	20	2	300	300	1/2
Concentric	2	2	200	200	2/3
Clouds	2	2	200	200	2/3
Waveform	21	2	300	300	1/2
Pendigitis	16	10	200	200	1/2
Magic	2	10	200	200	1/2
Yeast	8	10	200	200	2/3
Diabetes	8	2	200	200	2/3
Heart	13	2	200	200	2/3
wdbc	30	2	200	200	2/3
Image	18	2	200	200	2/3
Satimage	36	6	200	200	2/3

Figure 3.3 shows the bias and variance curves of αRSC in relation to α. We notice that increasing the values of α above 1 and up to the optimal value decreases the net variance.

[1]www.elena.com

We also notice the same decrease of unbiased variance. For biased variance, we notice both decrease and increase. As we learned from Domingos bias and variance decomposition in Chapter 2, high biased variance helps increase the overall error. Net variance may increases for higher values of α which result in a substantial increase in average loss. We conclude that pruning has a stronger influence reducing unbiased variance than biased variance. Increasing the values of α from 1 and to the optimal value may increases bias. For both Diabetes data set, bias decreases in relation to α. However, the bias on Diabetes data set increases for higher α values.

We conclude from these curves that pruning reduces unbiased variance resulting in a sharp decrease in overall error. On the other hand, pruning has limited influence in bias reduction. In fact, we notice a very high increase in bias for higher α values. This is not surprising, since pruning lowers the complexity of the decision boundary results in underfiting the data.

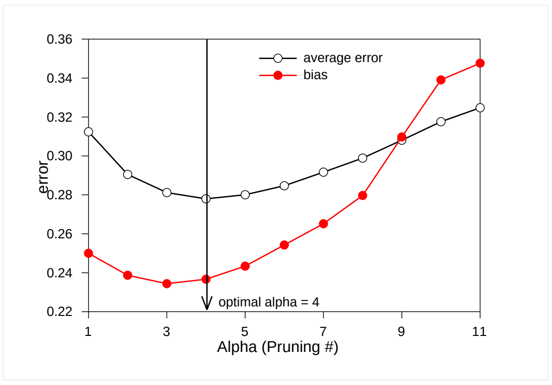
Figures 3.4 and 3.5 show the best results of the net variance and bias using unpruned versus pruned randomized sphere cover classifier on various data sets. We notice that pruning reduces significantly the net variance in comparison to unpruned classifier. This is not the case for bias, as we see a small decrease on only two data sets, Diabetes and Heart data sets, and small increase on the remaining data sets.

A closer look at the best pruned bias and variance results in comparison to unpruned results are shown in table 3.5, including unbiased and biased variance.
The results in table 3.5 show a pattern on the bias and variance performance of αRSC classifier:

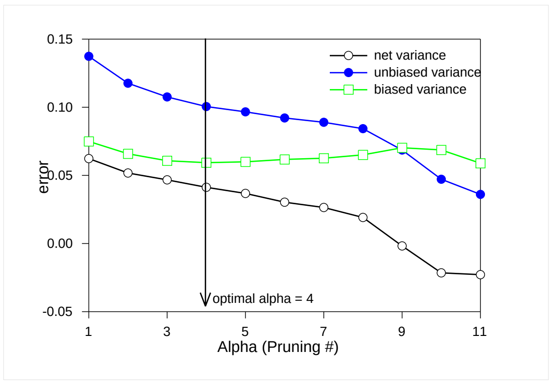
1. If pruning improves performance, which it does for the majority of cases in our experiment, we notice a substantial decrease in net variance. However, for these cases there are two trends in relation to bias.

    - Increase in bias is shown on Clouds (8.04%), Waveform (2.89%), Magic (10.62%), and wdbc (39.25%). However, unbiased variance is significantly decreased, as shown on Cloud (51.11%), Waveform (30.78%), Magic (57.16%), and wdbc (52.59%), which explains the decrease in average error.
    - Decrease in bias is shown on Diabetes (5.32%), Heart (9.06%), Twonorm (2.20%), and Yeast (0.43%). In all these cases both unbiased and biased variance is decreased.

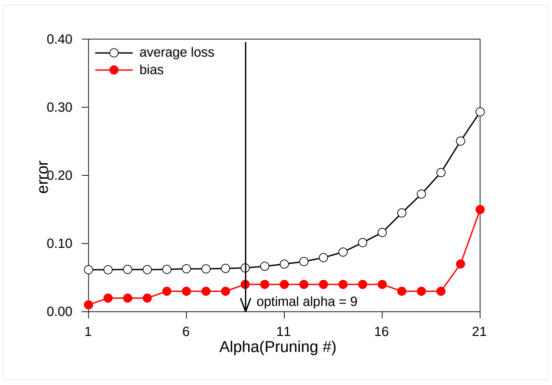
2. If pruning degrades performance, then we notice an increase in bias. This is shown on Pendigitis data set with 15.92% increase in average error for $\alpha = 2$, and a substantial



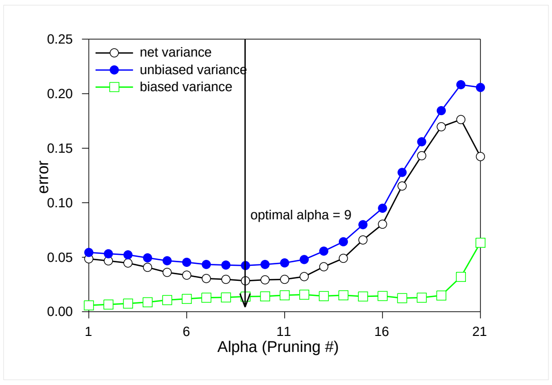
(a) Average error and bias curves of αRSC on Diabetes data set



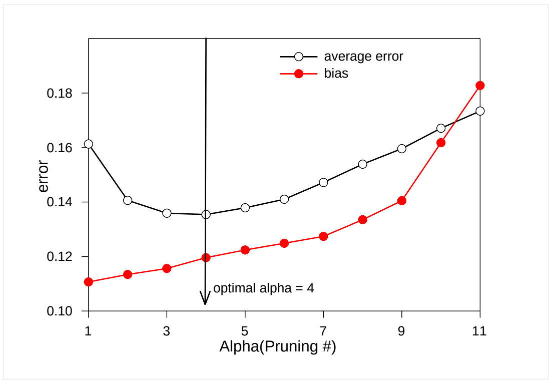
(b) Variance curves of αRSC on Diabetes data set



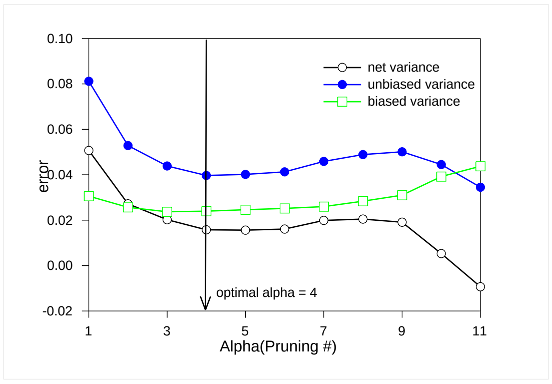
(c) Average error and bias curves of αRSC on Concentric data set



(d) Variance curves of αRSC on Concentric data set



(e) Average error and bias curves of αRSC on the Clouds data set



(f) Variance curves of αRSC on Clouds data set

Figure 3.3: Average error, Bias and Variance graphs of αRSC classifier in relation to α. The down arrows show the lowest error.

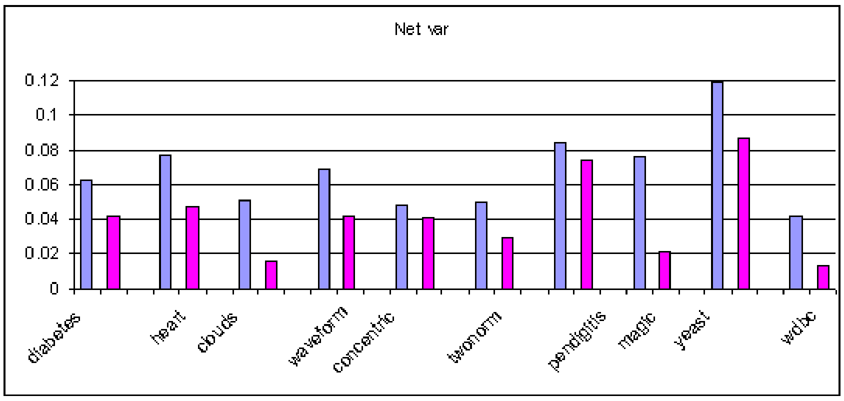


Figure 3.4: Comparing best net variance results of αRSC on various data sets. Pruning results are shown on the right hand side, and unpruned results are shown on the left hand side

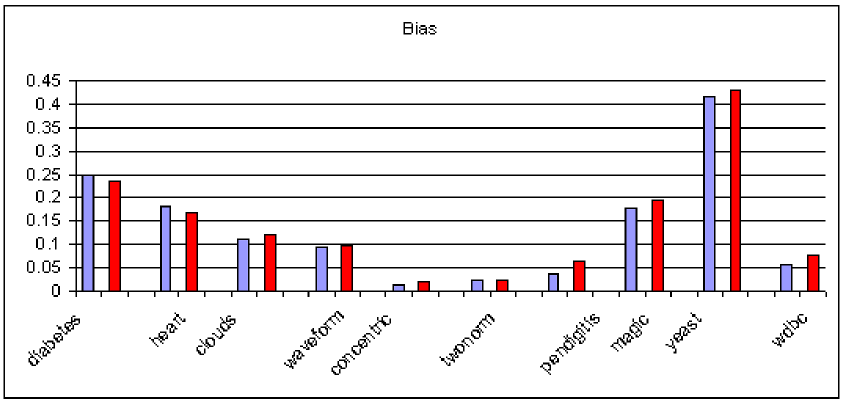


Figure 3.5: Comparing best bias results of αRSC on various data sets. Pruning results are shown on the right hand side, and unpruned results are shown on the left hand side

Table 3.5: Comparing best bias and variance results of αRSC on various data sets. Var. unb. and Var. bias. stand for unbiased and biased variance. Diff stands for the percentage difference between the pruned and unpruned values. The up arrow ↑ means an increase while a down arrow ↓ means a decrease.

Dataset	*Avg Error*	*Bias*	*Net Var*	*Var. Unb.*	*Var. bias.*
Diabetes					
$\alpha = 0$	*0.3124*	*0.2500*	*0.0624*	*0.1374*	*0.0750*
$\alpha = 3$	*0.2780*	*0.2367*	*0.0413*	*0.1006*	*0.0594*
Diff %	↓ ***11.01***	↓ ***5.32***	↓ ***33.81***	↓ ***26.78***	↓ ***20.80***
Heart					
$\alpha = 0$	*0.2599*	*0.1833*	*0.0765*	*0.1274*	*0.0509*
$\alpha = 7$	*0.2138*	*0.1667*	*0.0471*	*0.0872*	*0.0400*
Diff %	↓ ***17.74***	↓ ***9.06***	↓ ***38.43***	↓ ***31.55***	↓ ***21.41***
Clouds					
$\alpha = 0$	*0.1613*	*0.1107*	*0.0507*	*0.0812*	*0.0306*
$\alpha = 3$	*0.1354*	*0.1196*	*0.0158*	*0.0397*	*0.0240*
Diff %	↓ ***16.06***	↑ ***8.04***	↓ ***68.84***	↓ ***51.11***	↓ ***21.57***
Waveform					
$\alpha = 0$	*0.1626*	*0.0934*	*0.0692*	*0.1043*	*0.0352*
$\alpha = 11$	*0.1387*	*0.0961*	*0.0426*	*0.0722*	*0.0296*
Diff %	↓ ***14.70***	↑ ***2.89***	↓ ***38.44***	↓ ***30.78***	↓ ***15.91***
Concentric					
$\alpha = 0$	*0.0616*	*0.0131*	*0.0485*	*0.0544*	*0.0059*
$\alpha = 5$	*0.0630*	*0.0295*	*0.0335*	*0.0453*	*0.0118*
Diff %	↑ ***2.27***	↑ ***125.19***	↓ ***30.93***	↓ ***16.73***	↑ ***100***
Twonorm					
$\alpha = 0$	*0.0730*	*0.0227*	*0.0504*	*0.0586*	*0.0082*
$\alpha = 10$	*0.0515*	*0.0222*	*0.0293*	*0.0366*	*0.0073*
Diff %	↓ ***29.45***	↓ ***2.20***	↓ ***41.86***	↓ ***37.54***	↓ ***10.97***
Pendigitis					
$\alpha = 0$	*0.1206*	*0.0355*	*0.0850*	*0.0956*	*0.0106*
$\alpha = 1$	*0.1398*	*0.0652*	*0.0745*	*0.0913*	*0.0167*
Diff %	↑ ***15.92***	↑ ***83.66***	↓ ***12.35***	↓ ***4.50***	↑ ***100***
Magic					
$\alpha = 0$	*0.2510*	*0.1751*	*0.0759*	*0.1298*	*0.0539*
$\alpha = 4$	*0.2151*	*0.1937*	*0.0215*	*0.0556*	*0.0341*
Diff %	↓ ***14.30***	↑ ***10.62***	↓ ***71.67***	↓ ***57.16***	↓ ***36.73***
Yeast					
$\alpha = 0$	*0.5360*	*0.4170*	*0.1190*	*0.2045*	*0.0855*
$\alpha = 1$	*0.5159*	*0.4152*	*0.1007*	*0.1776*	*0.0768*
Diff %	↓ ***3.75***	↓ ***0.43***	↓ ***15.38***	↓ ***10.17***	↓ ***10.01***
wdbc					
$\alpha = 0$	*0.0978*	*0.0563*	*0.0415*	*0.0580*	*0.0165*
$\alpha = 8$	*0.0898*	*0.0784*	*0.0114*	*0.0275*	*0.0161*
Diff %	↓ ***8.18***	↑ ***39.25***	↓ ***72.53***	↓ ***52.59***	↓ ***2.42***

increase in bias (83.66%). Pruning, for this data set, has also increased substantially the biased variance (100%). Similarly, Concentric data set shows an increase in average error (2.27%) and a massive increase in bias (125.10%). We notice also a big increase for biased variance (100%). This should not be a surprise because Concentric data set is rather unusual, removing spheres that are close to the decision boundary will significantly underfit the data because no seperation exist between the two classes. As for Pendigitis data set, it is made of 10 classes which could be an issue for chosing the same α values for each class. Obviously, for both Concentric and Pendigitis data sets we see a decrease in net variance caused by the decrease in unbiased variance which emphasises the role of pruning in reducing the net variance.

The important observation that can be made from the bias/variance results is that pruning significantly reduces unbiased variance. However, in only few cases do we notice a small decrease in bias. Therefore, the decrease of αRSC average error in caused mainly by the decrease of unbiased variance.

3.3.5 Experiment 4: Comparing Classifier Accuracy Results

Table 3.6 shows the results of classification experiments using train/test split. Table 3.6 shows that αRSC has the second highest average rank of the five classifiers tested. KNN has the highest number of top ranks but on some datasets it performed relatively badly. These results suggest that αRSC can perform well on a variety of datasets in comparison to other classifiers, and that the smoothing process reduces the tendency of αRSC to perform very badly on some data sets. This is consistent with our observation that pruning decreases the unbiased variance. Further statistical tests performed on the Friedman averages showed that our first hypothesis that the classifiers average ranks are equal H0 is rejected which prompted us to perform post-hoc tests on pairs of classifiers. The post-hoc results showed that αRSC performs statistically better than decision trees and NNge. However, KNN showed no significant difference to αRSC. The post-hoc analysis demonstrate that αRSC is a promising classifier that can be used for many real world applications. The decrease of the generalisation error of αRSC is mainly attributed to the decrease of unbiased variance as it was fully explored in the bias/variance decomposition of Section 3.3.4.2.

Table 3.6: Classification accuracy (in %) and standard deviation of five classifiers in comparison with αRSC. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

data sets	αRSC	DT	K-NN	NB	NBT	NNge
sonar	83.43 ± 5.37	73.52 ± 5.63	85.57 ± 4.11	73.38 ± 4.91	74.14 ± 3.96	58.29 ± 4.48
heart	78.85 ± 3.62	77.19 ± 5.52	81.56 ± 2.75	85.11 ± 3.12	80.48 ± 3.70	78.74 ± 3.66
Haberman	73.37 ± 0.72	70.98 ± 4.19	74.44 ± 2.62	73.95 ± 2.32	72.61 ± 3.27	67.25 ± 3.91
cancer	70.93 ± 1.89	69.82 ± 6.76	74.77 ± 3.22	75.05 ± 3.25	74.52 ± 3.16	68.03 ± 5.15
ecoli	71.13 ± 2.50	81.28 ± 3.30	85.80 ± 2.78	85.33 ± 2.91	81.96 ± 2.76	83.78 ± 2.96
liver	60.90 ± 4.44	63.88 ± 4.37	62.32 ± 3.83	64.41 ± 4.01	63.71 ± 4.14	61.48 ± 5.01
iono	93.19 ± 1.46	75.05 ± 2.45	86.87 ± 2.58	91.99 ± 2.17	89.52 ± 1.72	91.23 ± 2.98
wdbc	92.33 ± 1.93	93.49 ± 2.05	95.11 ± 1.74	89.33 ± 5.52	93.79 ± 1.63	91.96 ± 2.91
wins	96.65 ± 1.10	94.03 ± 1.22	96.49 ± 0.57	97.18 ± 0.77	96.14 ± 1.08	96.01 ± 1.16
diabetes	74.09 ± 2.40	72.77 ± 2.55	74.66 ± 1.95	75.55 ± 1.88	73.87 ± 2.29	72.79 ± 2.28
vehicle	67.32 ± 1.93	70.91 ± 2.94	68.44 ± 1.50	58.96 ± 2.56	68.00 ± 2.06	61.81 ± 4.86
vowel	76.32 ± 1.60	74.54 ± 2.06	97.45 ± 1.09	66.37 ± 3.11	75.63 ± 3.06	83.40 ± 2.68
german	72.29 ± 1.70	70.68 ± 1.97	74.72 ± 1.64	71.16 ± 1.06	72.48 ± 2.32	63.29 ± 9.09
yeast	55.82 ± 2.52	53.44 ± 1.52	57.01 ± 1.78	57.95 ± 2.16	56.33 ± 1.92	52.77 ± 2.93
image	95.48 ± 0.66	93.94 ± 0.86	96.99 ± 0.54	78.40 ± 1.84	93.37 ± 0.94	86.46 ± 2.65
concentric	98.01 ± 0.57	98.42 ± 0.31	98.51 ± 0.32	98.19 ± 0.27	98.51 ± 0.27	89.72 ± 7.90
abalone	53.86 ± 1.09	51.67 ± 1.39	54.20 ± 1.16	52.13 ± 0.99	53.73 ± 1.44	50.51 ± 1.75
Clouds	88.50 ± 0.75	88.29 ± 0.56	88.62 ± 0.50	86.24 ± 0.51	88.51 ± 0.59	83.22 ± 1.02
waveform	89.31 ± 0.62	84.93 ± 0.64	89.64 ± 0.54	85.19 ± 0.65	88.12 ± 0.93	78.44 ± 3.73
banana	89.93 ± 0.43	88.78 ± 0.64	89.83 ± 0.66	72.51 ± 0.95	88.82 ± 0.76	82.67 ± 5.53
phono	87.35 ± 0.63	85.58 ± 0.80	88.86 ± 0.45	78.29 ± 0.74	84.20 ± 1.09	81.81 ± 1.60
satimage	90.21 ± 0.52	85.60 ± 0.62	90.55 ± 0.40	82.00 ± 0.69	82.43 ± 1.48	86.75 ± 0.90
twonorm	96.67 ± 0.46	84.35 ± 0.74	97.27 ± 0.31	97.53 ± 0.32	93.74 ± 1.63	79.04 ± 1.47
ringnorm	95.41 ± 0.40	90.82 ± 0.51	73.94 ± 0.62	98.44 ± 0.19	96.77 ± 0.66	91.62 ± 1.18
pend	98.19 ± 0.20	95.71 ± 0.30	99.08 ± 0.16	85.41 ± 0.46	94.34 ± 0.58	95.69 ± 0.65
magic	83.63 ± 0.37	84.63 ± 0.38	83.56 ± 0.34	75.80 ± 0.42	85.09 ± 0.33	81.60 ± 0.59
Average Ranks	3.04	4.23	1.94	3.62	3.21	4.96
Ranks	2	5	1	4	3	6

3.4 Chapter Summary

The sphere cover algorithm has been frequently proposed as a classifier [4, 63, and references therein]. The main research issue relates to the construction of spheres and their positioning to cover the training set. Indeed, this still the main focus for many researchers using the popular kernel methods [49, 50, 83, 136]. However, our first inquiry leads us to revise various sphere cover algorithms in order to find the best way to make them usable for ensemble methods. Our first investigation resulted in the randomized sphere cover classifier (αRSC). We investigated pruning and demonstrated with several data sets that it improves significantly the performance of this simple classifier. Using our pruning method, we showed that αRSC compares favorably with several known classifiers on many data sets. The second factor we looked at is the influence bias and variance have on the generalisation error of αRSC classifier. Our experimentations with the bias/variance decomposition of the error showed that pruning indeed reduce the average overall error. This reduction was mainly caused by the reduction of unbiased variance.

We summarize below the factors behind the choice of the sphere cover classifier as base classifier for our proposed ensemble methods of Chapter 5:

1. The geometrical property: spheres approximate the decision boundary by finding locally the boundary examples of each class. This approximation could potentially help us build diverse classifiers needed for the ensemble.

2. The training accuracy: Learning in classification depends on the capacity of the learner to produce a good hypothesis. Covering the entire training set will always result in a consistent hypothesis i.e. no training errors.

3. The low number of free parameters: the accuracy-complexity trade-off is regulated by a single parameter which takes a non-continuous values. This is a desirable property, since in classification a good hypothesis depends on the number of parameters used.

4. The flexibility of approximating the decision boundary: the sphere cover is flexible enough through its local spherical edges which should approximate various shapes of data set distribution.

Future research, will include an investigation into the possible use of a geometrical complexity analysis [61] of the data set prior to model selection for αRSC which may help giving rough estimates. There is no straight forward way to choose α in advance. However, in general, two factors should be considered for a good estimate:

1. The choice of the training set size. Large data set may require large pruning values. However, this is also dependent on the noise level.

2. The size and class overlap of the training set. Low class overlap may require small values for α and vice versa.

Chapter 4

Compression Scheme

4.1 Introduction

In this chapter, we analyse the proposed Randomized Sphere Cover classifier (αRSC) using compression scheme [1]. We verify that αRSC is a compression algorithm in order to use a data-dependent theoretical bound to analyse the components of αRSC that most influence its generalisation error. These components are the parameter α, errors made on the training set and the cardinality of a cover. The outcome of the compression scheme analysis prompted us to investigate whether reducing covering can be achieved better using a kernel method. To this end, in Section 4.3.3.1 we use two different datasets to evaluate αRSC with a Gaussian kernel.

From a machine learning perspective, analysing supervised learning algorithms is a very active research field with very long ramifications in statistical and mathematical theory [25, 36]. For example, in large margin classifiers statistical learning theory is repeatedly employed as a tool to explain their success [9, 56, 124]. In statistical learning theory, a classifier's performance on the training set (or test set) and the complexity of the hypothesis class are employed in probabilistic bounds [21, 133]. These probabilistic bounds are used to predict future errors of a learning algorithm, and they are called generalization error bounds (also known as risk bounds). Risk bounds basically provide upper (and possibly lower) ranges of the true error.

Compression scheme [38, 82] has been proposed to explain the generalization performance of **sparse algorithms**. In general, algorithms are called sparse because they keep a subset

[1] Note that compression scheme in classification is different to data compression theory. Compression scheme only interest is the reconstruction of labels. Conversely, the data compression theory is concerned with the reconstruction of the data itself.

from the training set as part of their learning process. A large number of algorithms fall in this category, such as Support Vector Machines (SVM) [56]. The SVM is an example of a sparse algorithm because the resultant classifier, the maximum-margin linear discrimination, depends uniquely on certain examples called support vectors. The liner function of an SVM can be reconstructed using only the support vectors without the need of the remaining data. The first to investigate the compression scheme was Littlestone and Warmuth using simple bounds [82]. Their findings explain that algorithms that possess these data compression characteristics alone are sufficient to guarantee learnability. However, it was not until 1995 that an extensive study was carried out by Floyd [38] which explored various bounds under the compression scheme with the intention of proving their effectiveness (also known as computational complexity analysis). In practice, compression bounds have been used for model selection instead of the cross validation technique [88, 89, 134].

Compression bounds make it clear that the answer to a good generalization performance of a classifier is data compression. However, data compression alone does not make necessarily an algorithm a compression algorithm. In Section 4.2 we define the compression scheme, and we show that αRSC is a compression algorithm. In chapter 3, we argued α controls the size of a cover as well as being the regularization parameter of choice. We also showed that α in most cases improves the classification error. Here, we examine the relationships between α, the accuracy and the cardinality of a cover using probabilistic bound based on the compression scheme. We show that manipulating these three quantities in the bound will result in accurate prediction of the true error. In addition, we investigate whether further compression is better achieved using kernel method.

4.2 The Compression Scheme

Littlestone and Warmuth [82] observed that for many algorithms only a subset of the training set is retained for classification. The basic idea is to have a compression algorithm that returns a subset of examples. A compression scheme for a concept class that consists of two functions comprises:

1. a compression function $\mathcal{C}$; and

2. a reconstruction function $\mathcal{R}$.

3. a message $\hat{\sigma}$ of additional information which is specific to the compression algorithm (we will give an example below of how to select $\hat{\sigma}$ for αRSC)

Therefore, the compression function takes as input a finite sample (set of examples) and outputs a subset of these examples. This subset is known as the **compression set**. The task of the reconstruction function is to use this compression set and (re)construct the hypothesis for the target concept to be learned.

Two important concepts used to study bounds in Statistical Learning Theory are Probably Approximately Correct (PAC) learning and Vapnik-Chervonenkis (VC) space dimension [56]. PAC learning was first introduced by Valiant [132] to show that learning of some unknown target concept, using a given hypothesis, requires an approximately correct answer with high probability in polynomial amount of time. This was shown as a relationship between the concept to learn, the size of the sample and the efficiency of the learning algorithm. The VC dimension of hypothesis space $\mathcal{H}$, is defined to be the maximum number d of examples that can be labelled as positive and negative examples in all 2^d possible ways, such that each labelling is consistent with some hypothesis in $\mathcal{H}$ [38, 55, 133]. Therefore, the VC framework uses VC-dimension as a measure of the complexity of the hypothesis space.

PAC Compression bounds are guarantees that for most training trials (draws of random training samples) the classification error of a classifier does not exceed an error threshold. A generalized theorem used for the compression algorithms in the PAC setting is described in [56]. A major aspect of this bound is that we can calculate the prediction bound of any data-dependent classifier [56]. It was shown in the past the difficulty of using theoretical bounds based on VC dimension for experimental evaluation [21, 106]. This is precisely the case for the sphere cover algorithm [21]. Cannon [21] explored an alternative approach in order to link it to non-compression data-dependent bounds with complicated and limited results. A full review of these bounds and their possible inter-links can be found in [106]. However, a much more straightforward method may be used based on the compression scheme which we describe here. Compression risk bounds found in the literature are too loose to be of any practical use, except that they show the desired relationship between sparse solution and generalization. A more general and tighter bound is proposed in [90] which is of practical use in our case. Before we move to the risk bound of described in [90], we need to show αRSC qualifies as a compression algorithm.

4.2.1 αRSC as a Compression Algorithm

The Class Cover Algorithms are a data compression since it is required that only a subset is kept for classification. αRSC algorithm can reconstruct the hypothesis (classifier) formed from the whole training set by using just a subset. Recall, a sphere is made of a (data-dependent) centre. Thus, the compression set will comprise centre examples. However, this will not give us the full information to get a specific sphere from a set of spheres with identical radii. We require to store either an information based on the radius of a sphere or store the border point in addition to the centre point. If we choose a method that requires us to identify spheres with their radii then we need to devise a method to differentiate between spheres with same radii. We will not use such method in this chapter as we show a simpler method may be employed since identifying a particular sphere requires just a centre and a border point. Therefore $\hat{\sigma}$ could be used in the bound below that reflect such choice. In both methods, the reconstruction function $\mathcal{R}$ simply consists of re-running the same αRSC algorithm with the compression set which consists of centre and border points and the additional information $\hat{\sigma}$ that identify each sphere. We obtain the same classifier as the one using the whole training set in S.

In the αRSC classifier, the entire compression set (centres + borders) is classified correctly as long as no pruning is performed ($\alpha = 1$). For classifiers that use the pruning parameter $\alpha > 1$, we include the examples that are mislabelled in order for the reconstruction to work. In this case, the compression set comprises centres and some uncovered border examples. It is required to include the information that this compression set admits some errors because of these uncovered border examples, since error are made on uncovered examples only. This method, however, will partially reconstruct the classifier. To fully reconstruct the classifier we require to explicitly state the uncovered examples which are not border examples. Thus, using solely information returned by the classifier on centres, borders and uncovered examples, we are able to reconstruct any hypothesis (lossless and lossy) in αRSC.

It is not required in the bound to specifically code the classifier and $\hat{\sigma}$. However, the bound will work only for compression algorithms as defined by the compression scheme.

4.2.2 A Sample Compression Risk Bound

An example $\mathbf{z} \overset{def}{=} (\mathbf{x}, y)$ is input output pair where $\mathbf{x} \in \mathcal{X}$ and $y \in C$. The compression algorithms have the following property. Given a training set $S = (\mathbf{z}_1, ..., \mathbf{z}_m)$, containing m examples where $m \in \mathbb{N}$. A learning algorithm $\mathcal{A}(S)$ which learns by compressing the data returns a classifier h that is identifiable by a subset $\mathbf{z_i}$ of S which is the compression set and a message $\hat{\sigma}$ of additional information which is specific to the compression algorithm and needed to obtain a classifier from the compression set $\mathbf{z_i}$ (we will give an example below of how to select $\hat{\sigma}$ for αRSC). Given a training set S, the compression set $\mathbf{z_i}$ is defined by a vector $\mathbf{i}$ of indices:

$$\mathbf{i} \overset{def}{=} (i_1, i_2, ..., i_{|\mathbf{i}|}) \tag{4.1}$$

$$\text{with} : i_j \in (1, ..., m) \quad \forall j \quad \text{and} : i_1 < i_2 < ... < i_{|\mathbf{i}|},$$

and where $|\mathbf{i}|$ denotes the number of indices present in $\mathbf{i}$. Hence, $\mathbf{z}_i$ denotes the i^{th} example of S whereas $\mathbf{z_i}$ denotes the subset of examples of S that are pointed by the vector of indices $\mathbf{i}$ defined above. To complete this notation we may have a set of indices $\bar{\mathbf{i}}$ not present in $\mathbf{i}$ such as $S = \mathbf{z_i} \cup \mathbf{z_{\bar{i}}}$. Therefore, the fact that any classifier returned by algorithm $\mathcal{A}$ is described by a compression set and message string implies that there exists a reconstruction function $\mathcal{R}$, associated with $\mathcal{A}$, that output a classifier $\mathcal{R}(\hat{\sigma}, \mathbf{z_i})$ when given an arbitrary compression set $\mathbf{z_i} \subseteq S$ and message string $\hat{\sigma}$ chosen from the set $\mathcal{M}(\hat{\sigma})$ of all distinct messages that can be supplied to $\mathcal{R}$ with the compression set $\mathbf{z_i}$. It is only when such a $\mathcal{R}$ exists that the classifier returned by $\mathcal{A}(S)$ is always identified by a compression set $\mathbf{z_i}$ and a message string $\hat{\sigma}$.

Therefore, a compression function $\mathcal{C}$ associated to $\mathcal{A}$ such as for every sample $S = (\mathbf{z}_1, ..., \mathbf{z}_m) \in \mathcal{X}$ we have $(\hat{\sigma}, \mathbf{z_i}) = \mathcal{C}(\mathbf{z}_1, ..., \mathbf{z}_m)$; and there exist a reconstruction function $\mathcal{R}$ which gives the classifier $\mathcal{A}(S)$ from $(\hat{\sigma}, \mathbf{z_i})$, i.e., $\mathcal{R}(\hat{\sigma}, \mathbf{z_i})$; such that:

$$\mathcal{A}(S) = \mathcal{R}(\mathcal{C}(S)) \;\forall\;\; S$$

The risk bounds[1] 4.3 and 4.4 are for arbitrary reconstruction functions that holds uniformly for all compression sets and message strings under the PAC settings. That is, each example $\mathbf{z}$ is drawn according to a fixed, but unknown, probability distribution $\mathbf{P}$ on $\mathcal{X}$, and the risk $R(h)$ of any classifier h is defined as the possibility that is mislabelled an example drawn according to $\mathbf{P}$.

We need to specify $\mathcal{M}(\hat{\sigma})$. We restrict the set $\mathcal{M}$ of possible messages $\hat{\sigma}$ to be a finite set which is dependent of the size of the compression set $\mathbf{z_i}$. The compression set size $\mathbf{z_i}$ will always be less or equal to the training set size S, and $|\mathbf{z_i}|$ must be far smaller than $|S|$ for the bounds to work.

[1]A full proof is given in [90]

$$\mathcal{M}(\hat{\sigma}) = \frac{|\mathbf{z_i}|}{|S|} \quad \forall \hat{\sigma} \in \mathcal{M} \tag{4.2}$$

For any given value to δ, which is between 0 and 1 (δ is normally chosen in order to give high probability), the error ϵ is less or equal the true risk R of compression learning algorithm with a reconstruction function $\mathcal{R}$ using both the compression set $\mathbf{z_i}$ and the addition information $\hat{\sigma}$. The error ϵ is calculated for a specific choice of $|\mathbf{i}|$, $|\mathbf{j}|$,and δ. $|\mathbf{j}|$ represents the number of errors made on the examples that do not belong to the compression set.

$$\epsilon(|\mathbf{i}|, \delta) \leq \frac{1}{m - |\mathbf{i}|} + \left[\ln \binom{m}{|\mathbf{i}|} + \ln\left(\frac{\mathcal{M}(\hat{\sigma})}{\delta}\right) + \ln(m+1) \right] \quad \text{for } |\mathbf{j}| = 0 \tag{4.3}$$

$$\begin{aligned} \epsilon(|\mathbf{i}|, |\mathbf{j}|, \delta) \leq \frac{1}{m - |\mathbf{i}| - |\mathbf{j}|} \Bigg[& \ln \binom{m}{|\mathbf{i}|} + \ln \binom{m - |\mathbf{i}|}{|\mathbf{j}|} + \ln \left(\frac{\mathcal{M}(\hat{\sigma})}{\delta}\right) \\ & + 2\ln(m+1) \Bigg] \quad \text{for } |\mathbf{j}| \geq 0 \end{aligned} \tag{4.4}$$

The risk bounds 4.3 and 4.4 may be used for αRSC error predictions. The important aspect of equations 4.3 and 4.4, as described by its authors, is the risk bound increases when the amount $|\mathbf{j}|$ increases. $|\mathbf{j}|$, on the other hand, is independent of the amount of errors made on the compression set. The risk bound will generally be smaller for sample-compression learning algorithms that always return a classifier making no errors on the compression set (i.e. makes no error in the compression set $|\mathbf{i}|$). However, this constraint might force the learner to produce classifiers with larger compression sets. Thus, the best classifiers are those that will have small error on $|\mathbf{j}|$, the training examples not in the compression set, have small compression set $|\mathbf{i}|$ and having small $\mathcal{M}(\hat{\sigma})$.

Various experiments are carried out to examine data compression, training error, and pruning values of αRSC classifier. These are the quantities manipulated in the compression bounds 4.3 and 4.4 to predict the lowest classification error.

4.2.3 Data Compression, Accuracy and Uncovered Examples

For the risk bounds to hold, the size of a compression set must be far smaller than the size of the training set. Table 4.1 shows the averaged compression sizes in percentage of the best classification accuracy based on 10 runs using 10 fold Cross Validation (i.e 10x10CV) and 2 fold Cross Validation (i.e 10x2CV for Ringnorm, Twonorm, Magic, Satimage, and Pendigitis). Data compression (as calculated in Equation 3.2) takes into consideration only centre examples (The additional information needed to be stored is the border point or the radii, see Algorithm 4).

Table 4.1, shows the data compression is above 90% on 12 datasets and is above 80% on four datasets. These are a substantial decrease in comparison to the Nearset Neighbor classifier which stores the entire training set. In addition, we showed the standard deviation to demonstrate the degree of variations between experiments (classifiers). Indeed, we notice that αRSC consistently compresses the training set.

Table 4.1: αRSC best data compression in (%) of the best average classification accuracy results using 10x10CV and 10x2CV for Ringnorm, Twonorm, Magic, Satimage, and Pendigitis.

Dataset	*Comp %*	*Std dev*	*Dataset*	*Comp %*	*Std dev*
Winsconsin	95.97	0.46	*Clouds*	95.26	0.19
Cancer	93.09	0.80	*Waveform*	99.44	0.08
Ecoli	81.29	1.46	*Image*	89.96	0.36
Glass2	90.50	1.03	*Vehicle*	83.84	0.71
Glass6	75.65	1.42	*Concentric*	98.33	0.13
Haberman	91.41	0.82	*Abalone*	92.16	0.23
Heart	99.09	0.18	*Yeast*	90.86	0.37
Ionosphere	78.74	1.18	*Vowel*	79.01	0.55
Sonar	80.28	1.35	*Twonorm*	98.98	0.15
Thyroid	95.32	0.91	*Magic*	89.48	0.23
wdbc	92.91	0.90	*Ringnorm*	81.37	0.49
Diabetes	82.29	0.84	*Satimage*	92.33	0.30
German	89.30	0.57	*Pendigits*	94.24	0.20

Risk bounds derived from the compression scheme show that low generalization error is achieved for small compression set and low training errors. The training accuracy in the case of a pure and proper cover is 100%. However, as shown in the previous chapter, overfitting is most likely to happen in this case. This is confirmed once more in Table 4.2 where each result shows the average training accuracy of the best classification accuracy using 10x10CV. As expected, we notice a big variation between datasets. Severely noisy datasets would require more pruning than low noise datasets. For instance, Yeast dataset is known to be very noisy and hence requires severe pruning as shown from the averaged training accuracy (52.25%). These results indicate that for a classifier to return the best classification accuracy some examples will be mislabled. As explained earlier, the risk bound that returns the lowest error must have the lowest number of training errors possible that are not in the compression set.

That is, the bound seeks a trade-off between a small compression set and low training errors of examples not in the compression set.

Table 4.2: αRSC Training accuracy in (%) of the best average classification accuracy results using 10x10CV and 10x2CV for Ringnorm, Twonorm, Magic, Satimage, and Pendigitis.

Dataset	*Training accuracy*	*Std dev*	*Dataset*	*Training accuracy*	*Std dev*
Winsconsin	97.10	0.79	*Clouds*	88.72	0.48
Cancer	75.13	3.74	*Waveform*	77.45	1.19
Concentric	99.16	0.34	*Glass6*	85.54	2.71
Ecoli	88.66	1.75	*Image*	99.25	0.39
Glass2	98.75	0.47	*Vehicle*	73.27	4.89
Haberman	76.56	1.55	*Abalone*	52.12	6.15
Heart	68.51	6.55	*Yeast*	52.25	7.61
Ionosphere	95.43	3.19	*Vowel*	93.16	2.26
Sonar	90.47	3.46	*Twonorm*	88.40	1.6
Thyroid	93.91	1.98	*Magic*	86.85	1.78
wdbc	98.17	0.48	*Ringnorm*	88.11	4.09
Diabetes	85.32	1.35	*Satimage*	89.51	1.45
German	78.98	1.39	*Pendigits*	99.20	0.15

The average percentages of uncovered examples using the best classification accuracy results of αRSC are shown in Table 4.3. As we have shown the result of best 10x10CV accuracies in the previous tables, we show the same returned results for the percentage number of uncovered examples. We notice again substantial variations between datasets. For example, Concentric dataset has the lowest result while Liver dataset has the highest result. The most noticeable dataset is Heart dataset which can be compressed up to 2.21 spheres on average of 10 runs, and shows 156.46 uncovered examples out of 243 training examples. A similar result is reported for liver. Results of Table 4.3 show that some datasets require severe pruning in order to get the best classification accuracy. In general, highly overlapping class distributions will results in higher number of spheres while well separated datasets will not. Similarly, a skewed class distribution which is overlapping a more compact dataset will result in severely pruned class over the other. In this case, pruning occurs more often for one class, which results in under representation of one class over the other. This is a possible explanation given for the reasons pruning varies greatly from dataset to dataset. In order to minimize the error on examples not in the compression set, αRSC is required to closely differentiate the decision boundary of each class.

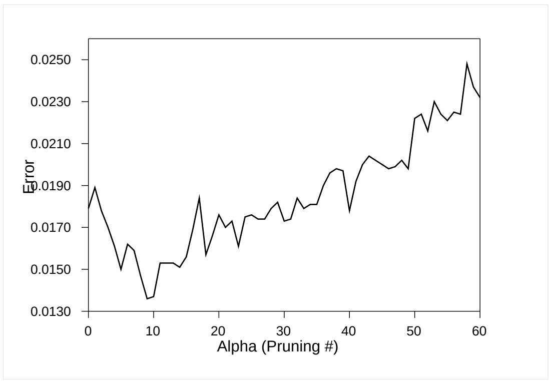
We plotted the errors of the risk bounds of Equation 4.3 for $\alpha = 0$ and Equation 4.4 for $\alpha > 0$. That is, instead of using the 10th partition for testing, we only use the prior information as returned by each αRSC classifier for various α values. These are the $|\mathbf{j}|$ quantity, the size of the compression set $\mathbf{z_i}$ and $\mathcal{M}(\hat{\sigma})$ as defined in Equation 4.2. The curves based on the risk bound (bound error) are shown on the right hand side of Figure 4.1 and 4.2 while the curves based on 10CV error (test error) are shown on the left hand side of Figure 4.1 and 4.2. It is interesting to note that the curves based on risk bounds are very similar to 10CV classification error. These experiments indicate that the information used in the risk bounds predict accurately the classification error as found by the 10CV.

4.2.4 Discussion

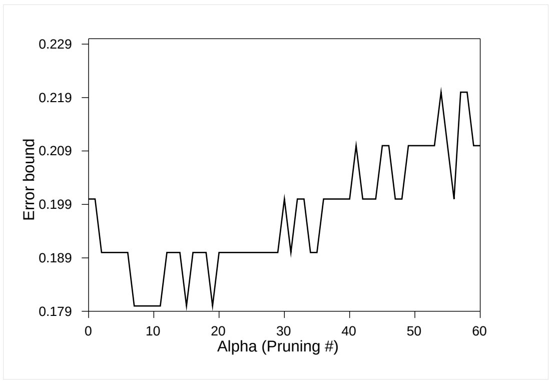
The generalization performance of αRSC classifier can be explained by the compression scheme framework. As we experimentally demonstrated, three factors in αRSC classifier are used in a compression bound to accurately predict the classification error. These three factors are: the compression set, which is the set of retained examples for classification; the pruning parameter which removes specific spheres from the cover; and the accuracy of the training set. We showed that these quantities can be used with the compression bounds described in Marchand and Sokolova [90] with no changes. The risk bound generated for set of α values are very similar to the 10-fold cross validation error on the same set of α values. This indicates that a model selection using the compression scheme is a valuable alternative to the common 10-fold cross validation as it is faster to execute. However, some important issues relating to compression risk bounds are not yet resolved. In fact, while writing this thesis, Hussain et al [64] outlined a number of drawbacks on many existing sample compression bounds for imbalanced dataset. Another area that we have not investigated has grown enormously in recent years is the study of PAC-Bayesian bounds [93, 94]. It has been shown that using the “Bayesian”approach to the already existing PAC bounds can tighten further the bounds. In addition, various existing Bayesian bounds have been all linked to a single compression lemma which proves the generality of the compression bounds [5]. This shows that compression bounds are an area of research that is constantly growing and could have a large application-base [81].

4.3 A Kernel Method for αRSC

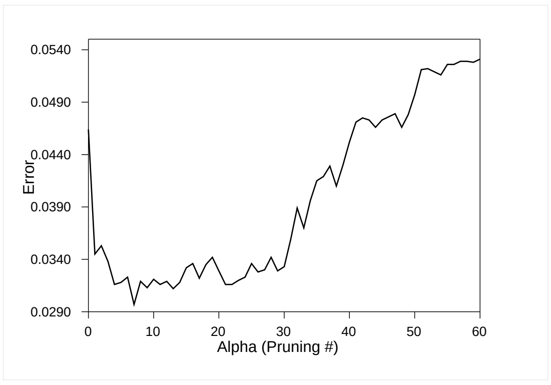
The previous experiments showed that a compression set can be represented with centre and border examples. The compression sets that have the lowest cardinality with fewest errors are preferred over larger consistent compression sets. In this case, if errors on examples not



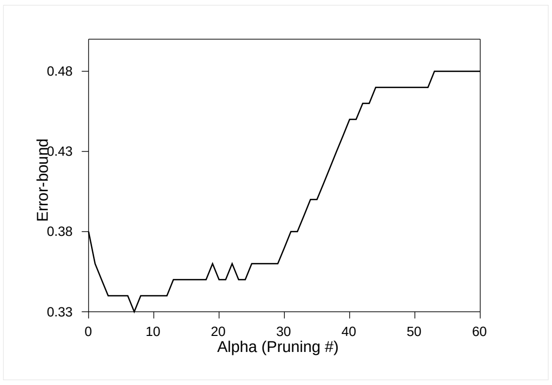
(a) αRSC test error curve of the Concentric dataset using 10CV



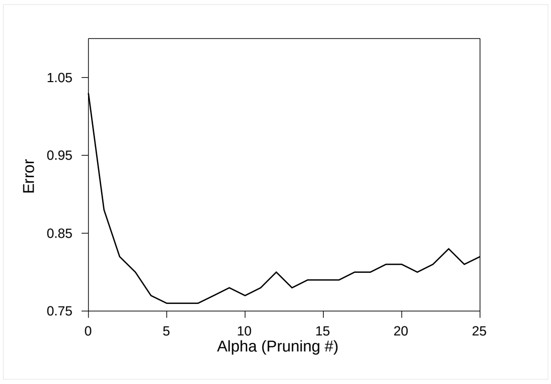
(b) αRSC bound error curve of the Concentric dataset



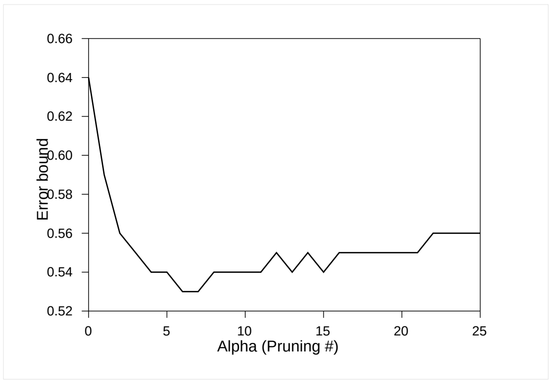
(c) αRSC test error curve of the Winsconsin dataset



(d) αRSC bounds error curve of the Winsconsin dataset

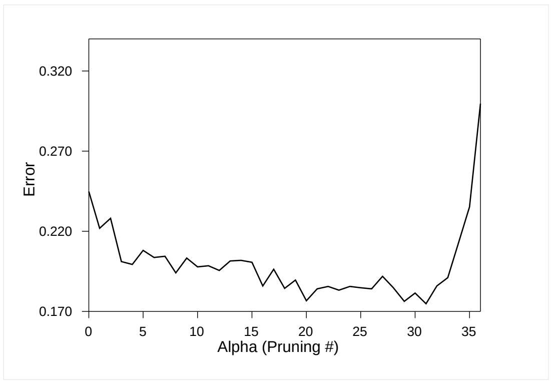


(e) αRSC test error curve of the Cloud dataset

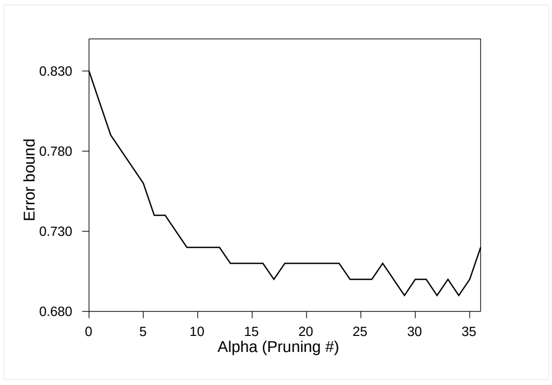


(f) αRSC bounds error curve of the Cloud dataset

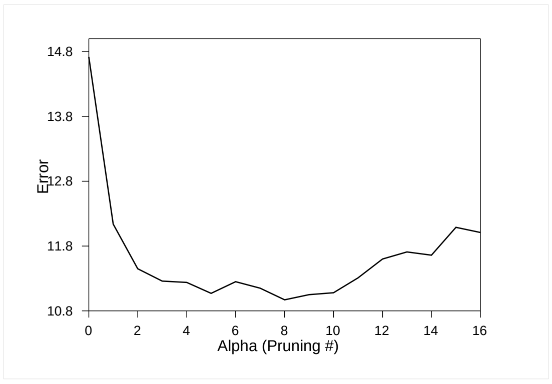
Figure 4.1: Learning curves based on 10CV classification error (on the left hand side) and the bounds error (on the right hand side) for αRSC classifier. We notice very similar curves for both 10CV and classification errors based on the compression bound indicating the compression bound gives very accurate estimate of the classification error.



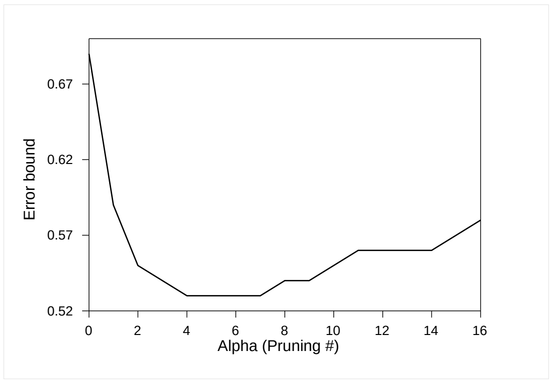
(a) αRSC test error curve of the Heart dataset using 10CV



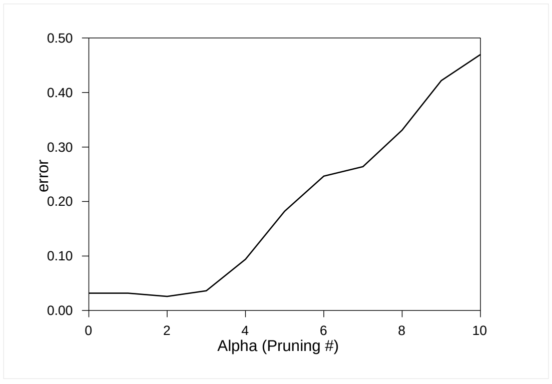
(b) αRSC bound error curve of the Heart dataset



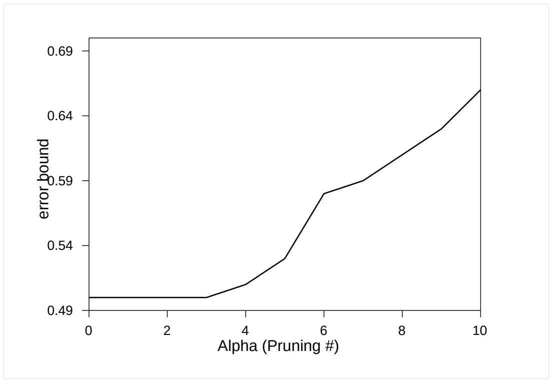
(c) αRSC test error curve of the Waveform dataset



(d) αRSC bounds error curve of the Waveform dataset



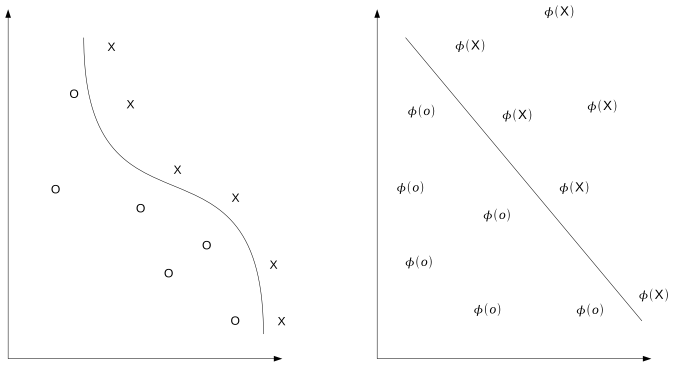
(e) αRSC test error curve of the Vowel dataset



(f) αRSC bounds error curve of the Vowel dataset

Figure 4.2: Learning curves based on 10CV classification error (on the left hand side) and the bounds error (on the right hand side) for αRSC classifier. We notice very similar curves for both 10 CV and errors based on the compression bound indicating the risk bound gives very accurate estimate of the 10 CV error.

included in the compression set are kept low we should end up with a very good classifier. The obvious next question is: Is it possible to improve this representation further in order to find the sparsest compression set with as low training error? In this section, we investigate one such possibility using Kernel methods.



(a) The feature Space (b) The Kernel Space after Transformation

Figure 4.3

4.3.1 A Useful Kernel Transformation

In this section, we take advantage of the easy transformation of a kernel matrix to an Euclidean distance matrix in order to run αRSC with minimum changes. The first step is to use a kernel function in order to get the kernel matrix. The Kernel methods [56, 124] map the data into some high-dimensional feature space (possibly infinite) F using:

$$\phi : \mathcal{X} \rightarrow F \quad \mathbf{x} \rightarrow \phi(x)$$

The advantage of such transformation is to have a separable dataset in the kernel space (called feature space) while it is not separable in the original space.

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle .$$

k : $\mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$. The main advantage of the Kernel trick is the possibility of doing inner product calculation in the feature space without worrying about the actual form of ϕ [56]. This mapping using the kernel function k is required to be positive definite (pd) [118]. Kernel matrix can be turned into a generalized distance measure using a simple transformation [118]. The distance $||\phi(x) - \phi(x')||^2$ in the feature space associated with a pd kernel k can be computed using the kernel trick as $k(x, x) + k(x', x') - 2k(x, x')$. This allows the distance to be expresses only by using the kernel, without explicitly performing the mapping. In this case, we get an Euclidean distance matrix from the mapping then simply construct αRSC classifiers.

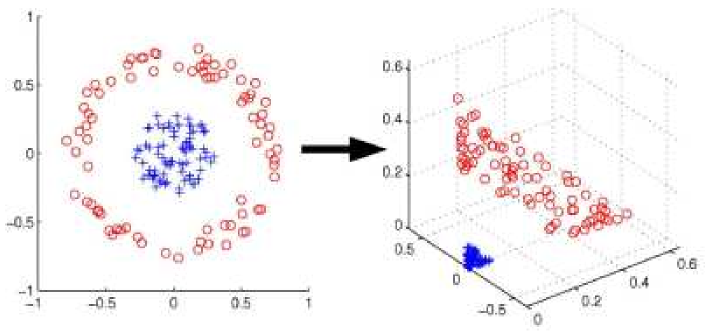


Figure 4.4: Two-dimensional toy example. The transformed input space using the kernel trick. An αRSC classifier constructed on kernel mapping may require a smaller compression set.

We keep the same parameter α in the kernalised αRSC ($\alpha KRSC$). We showed earlier that pruning reduces further the compression set and regularises complex hypothesis on noisy datasets. Therefore we are required to simply search the best mapping of the data into the feature space, with the chosen kernel function, which support the following arguments:

1. Small compression set $|\mathbf{i}|$.

2. Small error on $|\mathbf{j}|$, the training examples not in the compression set.

The reformulation of αRSC classifier can be described by two main criteria: a Kernel function with its set of parameters, and αRSC with the pruning parameter α. As shown in Figure 4.4, the class distribution of the transformed space could be well separated and nicely clustered which should help covering using fewer spheres than in the original space. In this case, the Kernel trick might solve both problems we outlined above. Our main goal is to search for a hypothesis that gives us the best trade-off between the compression set and the training error in the spirit of the compression scheme. The different steps of $\alpha KRSC$ are shown in Figure 4.5. To validate this proposition, we setup an experimental evaluation in the next section.

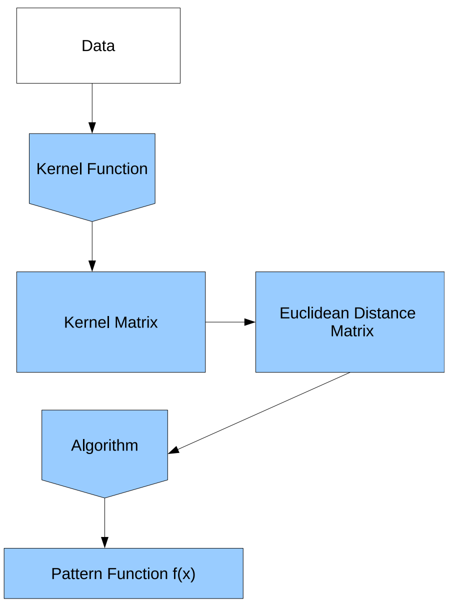


Figure 4.5: The different steps in $\alpha KRSC$. In the first step we get a kernel matrix with is then transformed into an Euclidean distance matrix

4.3.2 Experimental Setup

We showed in the previous sections that αRSC compression set consists of two subsets: each subset contains centre and border examples. In these experiments we wish to find out whether any of the two subsets (i.e. the centre set and border set) is reduced in size after the kernel transformation, and whether doing so improves or degrade classification error. For this empirical evaluation, we use the popular Gaussian RBF kernel:

$$k(x, x') = \exp\left(\frac{-||x - x'||^2}{\sigma}\right) \tag{4.5}$$

for varying values of the width parameters σ of the kernel

$$\sigma \in \{0.01, 0.02, 0.1, 0.2, 0.3, 0.5, 1, 2, 5, 10, 15, 20, 50\}$$

Depending of the choice of the kernel's width, we might either reduce the number of centre examples, the number of border examples or both. In order to verify this intuition, we employ the Concentric and Image dataset of Table 1.1. These two datasets are interesting for two different reasons. First, concentric dataset (1579 positive cases, 921 negative cases) is an artificial datasets that looks like two rings with no overlap between classes and no gap either, similar to figure 4.4. In the other hand, the Image dataset (1320 positives cases, 990 negative cases) requires a full cover in αRSC. Thus, increasing α degrades αRSC classification performance. We wish to find whether it will be "improved" by the Kernel transformation. We only show the best returned results from the above set of σ values. In addition, we use the compression bounds in order to verify whether it can select the optimal values.

4.3.3 Experimental Results

4.3.3.1 The Compression Factor in the Risk Bound

Table 4.4 shows the generalisation error results of $\alpha KRSC$ on the Concentric dataset for different Gaussian σ values. The aim is to find out whether the the lowest error achieved is on the most compressed set. In addition, we use the bound error to find out whether the compression risk bound of Section 4.2.2 can find this error in comparison to the 10CV error.

The results in Table 4.4 are from the average of 10 runs using 10 fold Cross Validation. The question we asked previously whether the kernel transformation helps to reduce the compression set is answered positively. The reduction in compression size is shown by comparing the original result in Table 4.4 (Concentric dataset), and the ones in the row where $\sigma = 1$. We notice the reduction of the size of the compression set comes from the reduction of both centre and border examples. A reduction from 75.37 [1] to 63.17 for the centers, and from 36.31 to 30.48 for the borders. The number of mislabled examples is is also reduced from 3.97 in the original result, to 3.88 for the Gaussian kernel. In the case where $\sigma = 2$, we notice a further reduction in both centers and border examples but at the expense of a slight increase in error. This reduction is from 75.37 to 57.89 for centers, and from 36.31 to 28.29 for border examples with a very small increase in mislabled examples, from 3.98 to 4.49 (about one or two misslabled examples for 100 experiments). Indeed, these results show that the kernel transformation works well for this dataset.

[1] We notice that this figure gives us a data reduction of 96.65% which is slightly higher than the value reported in Table 4.1 because $\alpha = 7$ for the selected risk bound.

Table 4.3: αRSC uncovered examples in % of the best average classification accuracy results using 10x10CV and 10x2CV for Ringnorm, Twonorm, Magic, Satimage, and Pendigitis.

Dataset	*% Uncovered examples*	*Std dev*	*Dataset*	*% Uncovered examples*	*Std dev*
Winsconsin	5.33	0.82	*Concentric*	1.56	0.33
Cancer	54.59	3.15	*Waveform*	41.65	1.29
Clouds	22.57	0.55	*Glass6*	18.88	1.81
Ecoli	13.51	1.26	*Image*	1.44	0.14
Glass2	2.47	0.69	*Vehicle*	31.77	1.12
Haberman	45.42	3.35	*Abalone*	64.87	0.77
Heart	64.39	4.67	*Yeast*	59.69	1.08
Ionosphere	6.91	0.72	*Vowel*	1.72	0.39
Sonar	17.85	1.82	*Twonorm*	22.19	2.58
Thyroid	9.62	1.93	*Magic*	23.49	0.94
wdbc	3.35	0.55	*Ringnorm*	13.33	0.63
Diabetes	27.67	1.28	*Satimage*	12.47	0.59
German	38.85	1.40	*Pendigits*	0.89	0.13

Table 4.4: Comparing best results using the Gaussian kernel on Concentric dataset. (error) stands for 10 CV error; (bound) stands for error found using the compression bounds; (centres) and (borders) are the number of centres and border examples that make up the compression set; (# incor) tr and (# incor) ts stands for the number of training and testing examples incorrectly labeled. (Org) stands for best results of αRSC (without kernel transformation).

α	7	5	9	17	8	14	18	16
Gaussian σ	**Org**	**0.01**	**0.02**	**0.1**	**0.5**	**1**	**2**	**5**
error	1.5919	2.504	2.0162	3.9795	1.56	1.55	1.79	3.5679
bound	0.199	0.4873	0.2896	0.1979	0.1987	0.1981	0.208	0.2618
centres	75.37	275.42	125.21	60.33	72.67	63.17	57.89	59.23
borders	36.31	36.9	39.85	29.18	34.82	30.48	28.29	29.92
# incor tr	15.399	19.0305	19.791	3.9795	17.19	24.2595	33.5205	60.54975
# incor ts	3.97975	6.26	5.0405	3.9795	3.91075	3.88025	4.49	8.91975

Table 4.5 shows the generalisation error results of $\alpha KRSC$ on the Image dataset for different Gaussian σ values. The aim is to find out whether the the lowest error achieved is on the most compressed set. In addition, we use the bound error to find out whether the compression risk bound of Section 4.2.2 can find this error in comparison to the 10CV error. The first two rows show the Gaussian parameter σ with two pruning values. Note that this dataset does not require pruning ($\alpha = 1$) to achieve lowest test error.

Table 4.5: Comparing best error results using the Gaussian kernels on Image dataset. (error) stands for 10 CV error; (bound) stands for error as found using the compression bounds; (centres) and (borders) are the number of centres and border examples that make up the compression set; (# incor) tr and (# incor) ts stands for the number of training and testing examples incorrectly labelled.

Gaussian σ	**5**		**1**		**10**		**15**	
α	**1**	**7**	**1**	**5**	**1**	**7**	**1**	**8**
error	4.4026	6.4113	4.3377	5.7359	3.8009	5.9913	3.7576	6.329
bound	0.5774	0.5445	0.5823	0.554	0.5786	0.5457	0.58	0.543
centers	373.84	214.93	377.85	232.07	374.81	201.63	375.94	200.96
borders	174.94	101.81	170.04	109.75	176.27	95.78	176.66	95.59
# incor tr	0	64.939644	0	57.249423	0	88.180785	0	73.989531
# incor ts	10.170006	14.810103	10.020087	13.249929	8.780079	13.839903	8.680056	14.61999

The results in Table 4.5 are from the average of 10 runs using 10 fold Cross Validation. From Table 4.5 (Image dataset), we notice some mixed results. That is, the Gaussian kernel transformation reduces the compression set. Yet, the bound fails to pick up the degradation found by the 10CV results. It is shown by comparing results for $\alpha = 1$ and where the bounds return the lowest error. For instance, for $\sigma = 15$, the reduction of the compression set is for both the centre and border examples. A reduction from 375.94 to 200.96 for the centres, and from 176.66 to 95.59 for the borders. The number of mislabelled examples is significant though 8.68 for the non pruned result and 14.62 for $\alpha = 8$. In this case, the bound returns the lowest value for $\alpha = 8$. However, for $\sigma = 1$, we notice a reduction in both centres and border examples but with a slight increase in error. This reduction is from 377.85 to 232.07 for centres, and from 170.04 to 109.75 for border examples with a very small increase in mislabelled examples, from 10.02 to 13.25.

4.4 Chapter Summary

In this Chapter, we described theoretical bounds and used them to analyse αRSC classifier. We answered several questions asked in this chapter. Section 4.2, demonstrates αRSC can be considered a compression algorithm following the compression scheme. In addition, we showed that α parameter plays a dual role in regularization and compression. Using the number of errors made on the training set, the size of compression set and some additional information to differentiate between classifiers, we showed how a risk bound can faithfully pinpoint the optimum value(s) for the pruning parameter α in the same fashion used for the 10 fold Cross Validation (CV) experiments. Thus, compression bounds can be a good alternative for model selection as it is substantially faster.

From the compression scheme perspective, we proposed a novel method using the kernel trick to manipulate the data rather than the spheres. This method is simplified by the way the Euclidean distance matrix is derived from the kernel matrix. In that case, we can build the classifier using the derived Euclidean matrix of the transformed space directly. The aim was to find a data distribution that gives us the sparsest possible set of spheres. Our initial experiments gave us mixed results. In the first case, we showed that our method reduces further the compression set, tightening in the process the compression bound. In the second case, the bound fails to pick up the best results. We believe the way α is used in αRSC classifier is at fault. That is, it does not take explicitly into consideration unbalanced datasets. It may be better to find different values for α on each class. This way we get a fair pruning strategy that will not over-prune or under-prune one class over the other. Nonetheless, for both dataset used in our kernel experimentation, we notice an improvement in reduction of the compression set (center and border examples).

For future research, we plan to find better representation of the compression bounds. In addition, time complexity analysis is required to find out whether a loss in learning speed is shown by building classifiers in the kernel space.

Chapter 5

Randomized Sphere Cover Ensembles

5.1 Introduction

In this chapter, we propose three ensemble methods for the Randomized Sphere Cover Classifier (αRSC). The first algorithm simply aggregates αRSC classifiers. The second algorithm, uses the β parameter, described in Chapter 2, to generate diverse covers for ensembles of αRSC classifiers. The final algorithm diversifies the ensemble by employing a random attribute selection for αRSC base classifiers.

Good ensemble design requires classifiers that are diverse. This is the key for improving classification accuracy [47, 74, 78, 125]. One way of diversifying αRSC ensembles is to allow spheres to cover a small number of examples from the opposite target class. In Chapter 2, Section 2.1.3.2 we described how in previous research, a parameter β representing the number of misclassified cases in a sphere has been used to make a sphere based classifier more robust by filtering outliers. In this Chapter, we empirically test whether we can use β parameter to diversify ensembles. In the CCP, both α and β parameters are chosen in advance, but β is harder to set, since ideally there should be a separate value for each sphere. However, it is generally infeasible to choose one value for each sphere in a single classifier. Here, we propose an automatic procedure as part of the proposed Randomized Sphere Ensemble ($\alpha\beta RSE$) which selects different values for β for different base classifiers. Thus, the aim of $\alpha\beta RSE$ is to perturb and aggregate covers while keeping accurate base classifiers. Optimal cover perturbation becomes essentially the action of tuning both α and β. We assess whether the proposed ensemble method $\alpha\beta RSE$ is different to an ensemble of αRSC classifiers without using β which we call it αRSE. In addition, we empirically compare the classification accuracy of both $\alpha\beta RSE$ and αRSE to those of several known ensemble methods.

In real world applications we are faced with large datasets in high dimensional space. For example, web mining, text categorization, financial forecasting, and gene expression profiling are some examples of domains in which huge amounts of information have to be employed. As such, classifying, understanding or compressing this information becomes a very difficult task. A solution that has been a very active research topic for many decades is called attribute selection[1] [97]. Attribute selection is recognized as an important process in data analysis because it speeds up the learning, improves the data quality, and increases the accuracy of the resulting model [52, 86, 138]. Furthermore, it is well established that many classification algorithms suffer from redundant, irrelevant or noisy attributes, specifically instance based learning using distance metrics [1, 77]. Attribute selection for ensemble learning is an active area with promising results [112, 128]. We described several methods in chapter 2. However, ensemble methods that randomly select attributes have been shown to work particularly well for several base classifiers and especially for nearest neighbour classifier (K-NN) [16, 59, 122]. Working with random subsets of attributes is termed the random subspace method [58]. In this chapter, we investigate using αRSC using random subsets of attributes. We call this third type of ensembles the Randomized Subspace Sphere Cover ($\alpha RSSE$).

It has been shown that aggregating classifiers essentially decreases variance [47]. This was shown for ensembles of decision trees [14, 47], SVM [131] and neural networks [46]. We showed, in chapter 3, that the pruning parameter α reduces the variance of the αRSC classifier resulting in overall loss reduction. Conversely, bias was not substantially decreased, and in some cases we observed an increase. Section 5.3.2.3 continues with the bias/variance experimentations on $\alpha\beta RSE$ and $\alpha RSSE$ in order to assess the generalization error of the different ensembles. We seek to answer the following question: Does an ensemble of randomised sphere cover classifiers reduce both bias and variance, or variance only? In addition, we explore the bias and variance terms of $\alpha RSSE$ in order to gain further insight into the classification error of $\alpha RSSE$.

[1]It is also known as feature selection, we choose not to use feature as it maybe confused with feature space of the kernel method

5.2 Ensemble Algorithms

5.2.1 αRSE

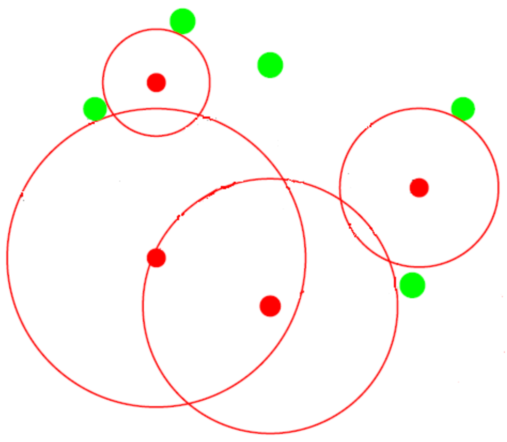
One of the basic design criteria for αRSC was to randomize the cover mechanism so that we could create diversity in an ensemble. Hence our first ensemble algorithm, αRSE, is simply a majority voting ensemble of αRSC classifiers. With all ensembles we denote the number of classifiers in the ensemble as L. We fix α for all members of the ensemble. Each classifier is built using the algorithm described in Algorithm 4 using the entire training data. The basic question we experimentally assess is whether the inherent randomness of αRSC provides enough implicit diversity to make the ensemble robust.

5.2.2 $\alpha\beta RSE$

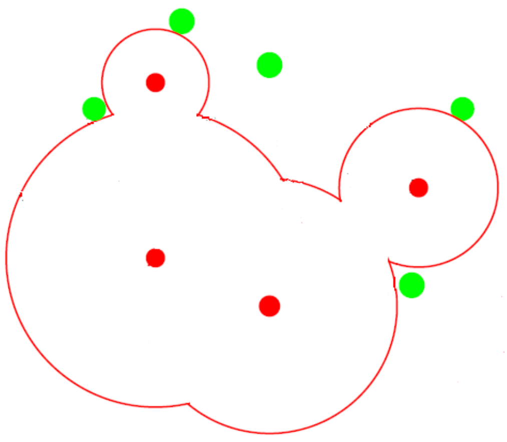
Finding "bad" examples in order to avoid over-training (or overfitting) is not trivial. Noisy examples are not known in advance and this makes the classification problem difficult. In classification, border examples are essential for approximating a decision boundary [15, 57, 61]. In most cases, the border examples are near the decision boundary except for those which may be outliers [26, 80]. We showed in Chapter 3 that the αRSC classifier uses the α parameter in order to find an approximately optimal decision boundary by using model selection. Generally, overfitting is avoided by choosing $\alpha > 0$ which may leave examples near the decision boundary uncovered. In overlapped datasets, some of the uncovered examples are noise. Identifying these noisy examples during learning should enhance the ensemble classification performance.

$\alpha\beta RSE$ uses border and uncovered examples in order to further randomize (perturb) covers. The aim of the $\alpha\beta RSE$ is to create diverse and accurate base classifiers. $\alpha\beta RSE$ uses the following process: To simplify the description of a training set from an αRSC point of view, we divide the training sample into three sets. Border examples are stored in set E, uncovered examples are stored in set F, and mislabelled examples are stored in set G. Recall, border example is defined as the example which is the closest to a sphere of different target class. Obviously, these sets are not mutually exclusive. The remaining examples that are not in any of these three sets are just called "data". A first base classifier is trained on a data set. Border, uncovered and mislabelled examples are identified. The second classifier is trained on a new training set with border examples in set E removed and replaced with randomly and uniformly sampled with replacement from set $E \cup F \cup G$. Consequently, fewer examples from the original training set might be mislabelled because omitting border examples allow spheres to grow into zones of different target class (an illustration is given in Figure 5.1). The process selects automatically values for β parameter which may be different for different spheres. β examples are the mislabelled examples stored in set G in addition to the mislabelled uncovered examples. The training process is repeated to create diverse base classifiers in the ensemble.

The first cover (classifier)

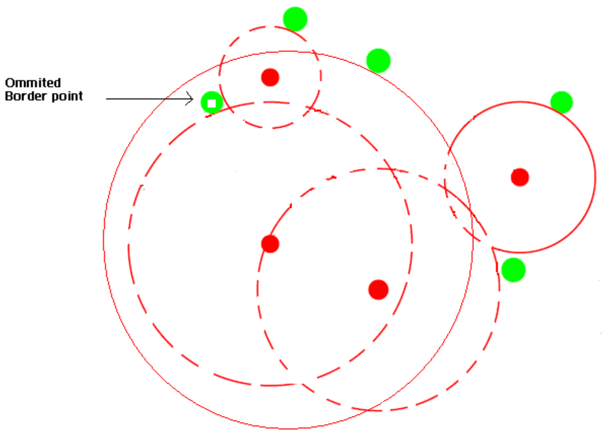


(a) A two class toy problem to illustrate the automatic selection of β in $\alpha\beta RSE$.

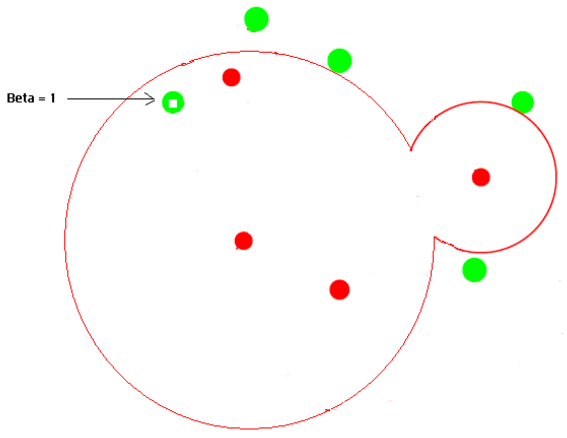


(b) The decision surface is the union of circles.

The second cover (classifier)



(c) A Border point is omitted in training resulting in $\beta = 1$ for this circle. Dotted cirlces represent previous cover.



(d) The new decision surface is the union of these circles.

Figure 5.1: An illustration showing a cover modification with β parameter on a binary class toy dataset.

The $\alpha\beta RSE$ algorithm use a predefined number of αRSC base classifiers, which have a single user defined integer parameter, α, that specifies the minimum size for any sphere. The ensemble also uses the parameter β, which is selected automatically during training and specifies the number of examples inside any such sphere of opposite target class. Informally, for any given α, $\alpha\beta RSE$ works as follows.

1. Repeat for a specific number of iterations (building classifiers for the ensemble)

   (a) Build a base classifier (cover) using the training set D.

   (b) Find the border points and store in set E.

   (c) Find the uncovered examples and store in set F.

   (d) Find the misclassified examples on the training set D and store in set G.

   (e) Classify the test examples and store the predictions.

   (f) Create set $U = E \cup F \cup G$.

   (g) Replace border points E in D by sampling from U randomly and uniformly.

2. Use the majority vote on stored prediction for each test example to get the ensemble prediction

In summary, $\alpha\beta RSE$ uses a uniform random sampling in order to replace border examples with mislabelled and uncovered examples of the previous classifier, thus taking advantage of the geometrical properties of the base classifier to select values for β parameter and randomize further the cover. An illustration is given, in Figure 5.1 (b), by the original decision boundary, and in Figure (d) by a newly and different created decision boundary. As such, the proposed ensemble method diversifies members and keeps accurate classifiers, and in the process should improve performance. In the next section we verify this assumption with an experimental evaluation. Note there may be overlap between the sets E, F and G but we do now allow duplicates in the training set.

A formal description is given in Algorithm 5. New cases are then classified by a majority vote of the L classifiers. The principle idea is that we replace the training data by removing border cases at every iteration of the ensemble construction. The data driven iterative approach adopted has strong analogies to constructive algorithms such as boosting, described in Section 2.4.2.

5.2.3 A Random Subspace Sphere Cover Ensemble, $\alpha RSSE$

The Random Subspace Sphere Cover ($\alpha RSSE$) Algorithm 6 builds base classifiers using random subsets of attributes by sampling without replacement from the original full attribute set. κ attributes are selected in the subset to train a base classifier. The same attribute might be selected again by another specific base classifier in the ensemble. Random covers in the subspace is the same as selecting examples with attributes that have no-zeros contribution to the Euclidean distance $\mathfrak{d}$. Each time a classifier is added to the ensemble, a subset of attributes is computed by randomly and uniformly selecting without replacement κ attributes from the full set of attributes. The ensemble generates base classifiers in the same subspaces using the same number of κ attributes. A test example $\mathbf{x}$ uses the same set of attributes for classification. Therefore, $\alpha RSSE$ combines outputs from multiple classifiers each having access only to a random subset of attributes. The majority vote is employed for classification.

The $\alpha RSSE$ ensemble can be compared with existing subspace methods. The random subspace ensemble constructs decision tree based classifiers that maintains highest accuracy on training data and improves on generalization accuracy as it grows in complexity. The ensemble consists of multiple trees constructed in randomly chosen subspaces. The popular Random Forests algorithm builds a tree using a bootstrap replica of the learning sample, and a decision tree without pruning. At each test node the optimal split is derived by searching a random subset of candidate attributes selected without replacement from the candidate attributes. Random forest combines randomization with bootstrap sampling. As explained earlier, base classifiers generated based on a permutation of attributes may have both distinct subsets of attributes that forms random cover, and randomized covers based on selecting random centres from the subset of attributes already selected but in different order. Therefore, our double randomization method might be compared usefully to random forest.

5.3 Experimental Evaluation

The accuracy results provided in this section are based on an independent test set drawn randomly from the data set. We use 2/3 of the data set for training and tested the classifiers on the same remaining test sets. We choose to use the average of 30 runs in order to make a fair comparison. We employ the same previous α values of Section 3.3.1 in αRSE and $\alpha\beta RSE$. For $\alpha RSSE$ we use 5 average runs on the training set alone to select the best set of parameters (α and κ). For comparison purposes we used Adaboost, Bagging, Random

Algorithm 5 A Randomised Sphere Cover Ensemble ($\alpha\beta RSE$)

Input: Cases $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, distance function $\mathfrak{d}(\mathbf{x}_i, \mathbf{x}_j)$ parameters α, L.

Output: L random sphere cover classifiers $B_1, \ldots, B_L$

```
1:  $D = D_1$
2:  for $j = 1$ to $L$ do
3:      $B_j =$ buildRSC($D_j, \alpha$).
4:      $E =$ borderCases($B_j, D_j$)
5:      $F =$ uncoveredCases($B_j, D_j$)
6:      $G =$ misclassifiedCases($B_j, D_j$)
7:      $U = E \bigcup F \bigcup G$
8:      $D_{j+1} = D_j - E$
9:      for $m = 1$ to $|E|$ do
10:         $C =$ randomSample($F$)
11:         $D_{j+1} = D_{j+1} \bigcup C$
12:     end for
13: end for
```

Algorithm 6 A Random Subspace Sphere Cover Ensemble ($\alpha RSSE$)

Input: Cases $D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, distance function $\mathfrak{d}(\mathbf{x}_i, \mathbf{x}_j)$ parameters α, L, κ.

Output: L random sphere cover classifiers $B_1, \ldots, B_L$ and associated attribute sets $\mathcal{K}_1, \ldots, \mathcal{K}_L$.

```
1: for $j = 1$ to $L$ do
2:     $\mathcal{K}_j =$ randomAttributes($D, \kappa$)
3:     $B_j =$ buildRSC($D_j, \alpha$)
4: end for
```

Committee and Multiboost. Adaboost and Bagging use C4.5 decision trees without pruning. Random Committee create a committee of random classifiers. The base classifier that forms the committee members is the random tree classifier which construct a tree that considers K randomly chosen attributes at each node. MultiBoosting [135] can be viewed as combining AdaBoost with wagging. Wagging can be considered as Bagging with allocation of weights from the Poisson distribution. MultiBoost uses the C4.5 classifier as the base learner. WEKA [143] implementations are used for the standard classifiers, bespoke implementation for our proposed ensembles which is in C++. The WEKA ensembles are trained using the default parameters in WEKA [108]. The ensemble sizes are 25 for the first set of experiments and 100 for the second set of experiments. Experiment 1 compares the accuracy of our proposed ensembles. Experiment 2 is the analysis of the proposed ensembles using the bias/variance decomposition. Experiment 3 compares classifiers accuracy results.

We use the Friedman test described in Chapter 3. Friedman test checks whether the measured average ranks are significantly different from the average rank. The null-hypothesis states that all algorithms are equivalent and their ranks should be equal. If the null hypothesis is rejected we proceed to a post-hoc pairwise test [67].

5.3.1 Experiment 1: Proposed Sphere Cover Ensembles

Table 5.1: Classification accuracy (in %) and standard deviation of $\alpha\beta RSE$, αRSE, $\alpha RSSE$, using average results of 30 different runs on the same independent train/test splits. Ensembles are trained using 25 base classifiers.

Data Set	αRSE	$\alpha\beta RSE$	$\alpha RSSE$
Abalone	54.25 ± 0.94	**54.89** ± 1.02	54.77 ± 1.28
waveform	90.40 ± 0.67	**90.68** ± 0.65	90.21 ± 0.51
satimage	90.90 ± 0.41	90.90 ± 0.41	**91.71** ± 0.47
ringnorm	96.71 ± 0.38	97.17 ± 0.30	**98.29** ± 0.26
twonorm	97.32 ± 0.26	**97.41** ± 0.26	97.03 ± 0.30
image	96.87 ± 0.50	96.87 ± 0.51	**97.39** ± 0.65
german	73.21 ± 1.76	74.00 ± 1.69	**74.59** ± 1.47
wdbc	93.21 ± 1.47	93.86 ± 1.52	**94.67** ± 1.33
yeast	56.34 ± 2.09	58.22 ± 1.24	**58.80** ± 1.90
diabetes	74.52 ± 1.78	75.01 ± 1.79	**76.17** ± 2.25
iono	93.48 ± 2.05	93.39 ± 2.25	**94.53** ± 1.79
sonar	**84.67** ± 4.17	84.43 ± 3.66	84.52 ± 4.49
heart	78.85 ± 3.60	80.74 ± 3.26	**82.74** ± 4.02
cancer	69.46 ± 2.97	70.07 ± 3.62	**76.27** ± 2.96
winsc	95.53 ± 1.34	95.67 ± 1.33	**97.21** ± 0.95
ecoli	85.36 ± 2.78	**85.51** ± 2.64	85.00 ± 2.07

The average classification results and standard deviation for $\alpha\beta RSE$, αRSE, and $\alpha RSSE$ using a size of 25 classifiers are summarized in Table 5.1.

Table 5.2: Classification accuracy (in %) and standard deviation of $\alpha\beta RSC$, αRSE, $\alpha RSSE$, using average results of 30 different runs on the same independent train/test splits. Ensembles are trained using 100 base classifiers.

Data Set	αRSE	$\alpha\beta RSE$	$\alpha RSSE$
Abalone	54.36 ± 1.16	54.48 ± 1.23	**54.91** ± 0.98
waveform	90.56 ± 0.70	90.32 ± 0.66	**90.73** ± 0.53
satimage	90.91 ± 0.38	91.12 ± 0.44	**91.92** ± 0.54
ringnorm	96.88 ± 0.37	97.54 ± 0.31	**98.43** ± 0.27
twonorm	97.36 ± 0.28	**97.49** ± 0.22	97.39 ± 0.28
image	96.77 ± 0.50	96.80 ± 0.56	**97.83** ± 0.53
german	73.23 ± 1.82	74.16 ± 1.58	**74.28** ± 1.56
wdbc	93.39 ± 1.56	93.91 ± 1.57	**95.00** ± 1.44
yeast	57.26 ± 1.44	58.41 ± 1.36	**59.43** ± 1.93
diabetes	74.53 ± 1.84	75.04 ± 2.57	**76.25** ± 2.21
iono	93.56 ± 2.06	93.53 ± 1.96	**94.76** ± 1.68
sonar	84.86 ± 4.23	85.00 ± 3.72	**85.24** ± 5.39
heart	79.26 ± 3.40	80.67 ± 3.10	**84.00** ± 3.43
cancer	69.53 ± 3.29	69.58 ± 3.32	**76.16** ± 2.75
winsc	95.54 ± 1.33	95.71 ± 1.33	**97.42** ± 0.91
ecoli	85.54 ± 2.96	**85.86** ± 2.65	85.71 ± 2.36

We also count a total increase in accuracy for $\alpha\beta RSE$ on 12 datasets in comparison to αRSE. Comparing $\alpha RSSE$ with $\alpha\beta RSE$ shows a total increase in accuracy on 11 datasets. $\alpha RSSE$ shows considerable improvements in comparison with both αRSE and $\alpha\beta RSE$ when the ensemble size is increased to 100 as shown in table 5.2. The result shows that the best results are achieved with $\alpha RSSE$. $\alpha\beta RSC$ is the second best.

It is not clear whether $\alpha RSSE$ achieves high accuracy on the training set. Our intuition, by looking at our previous results, is high training accuracies should depend on the chosen values given to α. Covering the entire subspace will still produce 100% accuracy. However, pruning also applies for the subspaces which should regularize complex hypothesis. Looking at Table 5.3, we notice that pruning was required on most datasets in order to achieve the best classification accuracy. It demonstrates that pruning is necessary for complex hypotheses built in the subspaces.

In addition, Table 5.3 shows the average training accuracy, the average classification accuracy, the ensemble training accuracy and the standard deviations for the 25 classifiers that are used to train $\alpha RSSE$. The maximum training accuracy for the base classifier and the ensemble is achieved on 4 datasets. The remaining datasets show varied training accuracies with the exception of Thyroid which achieved the maximum training accuracy with pruning. We notice a high average training accuracy, except for Yeast (45.52%) and Abalone (61.24%), which suggest that the base classifiers achieve high training accuracy in the subspace. The same table shows the average classification accuracies of the 25 base classifiers in order to compare them with those of a single αRSC classifier trained on the full attribute set. We notice that the classification accuracy of the base classifiers in $\alpha RSSE$ are not as accurate as those of a single αRSC classifier except for Glass2 and Ionosphere datasets. Therefore, we conclude that $\alpha RSSE$ requires relatively accurate base classifiers in order to while "boost" the classification accuracy of the ensemble.

Table 5.3: $\alpha RSSE$ Accuracy and standard deviation results in % using 25 base classifiers. (Avg Tr Acc) stands for average training accuracy, (Avg Ts Acc) stands for average test accuracy, and (Ens Tr Acc) stands for Ensemble Training Accuracy. These results are for those ensembles that returned the best classification accuracy using 10 fold Cross validation for the average of 10 runs (i.e.average of 100 accuracy results (10x10CV)) and 2 fold Cross-Validation for Twonorm, Ringnorm and Satimage (i.e.average of 20 accuracy results (10x2CV))

Dataset	*Avg Tr Acc*	*Std dev*	*Avg Ts Acc*	*Std dev*	*Ens Tr Accu*	αRSC
Image	100	0	94.64	1.96	100	96.1
Yeast	45.52	8.7	51.33	4.95	72.73	57.48
Abalone	61.24	5.09	53.52	1.85	76.04	54.44
Waveform	95.34	0.59	83.9	1.9	99.79	89.56
Twonorm	97.69	0.25	90.2	0.46	99.98	96.59
Satimage	99.99	0.01	84.58	1.33	100	88.95
Ringnorm	100	0	92.48	0.42	100	95.6
Ecoli	87.63	2.09	82.21	4.05	91.59	85.09
Cancer	77.21	6.42	72.38	5.59	80.11	74.4
Wins	84.54	18.84	93.92	2.77	99.36	97.03
wdbc	98.92	0.47	95.29	2.09	99.65	96.26
German	83.93	2.26	72.65	2.86	84.43	73.87
Diabetes	77.09	2.6	73.48	4.08	77.95	74.63
Ionosphere	100	0	92.01	3.5	100	93.4
Heart	80.94	3.53	75.43	7.19	94.67	82.81
Sonar	100	0	75.48	8.83	100	82.8

5.3.2 Experiment 2: Analysis of $\alpha\beta RSE$ and $\alpha RSSE$

5.3.2.1 Learning Curves of $\alpha\beta RSE$ and $\alpha RSSE$

Figure 5.2 and 5.3 show the graphs of the classification accuracy of $\alpha\beta RSE$ in relation to pruning parameter α for four different datasets. The learning curves are for ensembles made of 25 base classifiers (the classification average accuracy of 10 different runs). Furthermore, we show, in the same figures, the accuracy curves of the 25 averaged classifiers using the same 10 runs. We notice a common result on each dataset. That is, the ensemble 10CV classification accuracies are better than those of the 25 averaged classifiers. These results confirm that the proposed ensemble improves the classification accuracy of single classifiers. In addition, we notice both curves follow a similar evolution in relation to α. That is, α values that returned the best classification accuracy for $\alpha\beta RSE$ are similar to those of a single classifier. However, we notice that $\alpha\beta RSE$ is less sensitive to α parameter indicating members of the ensemble complement each other. An example is given in Figure 5.4. Two learning curves representing the classification error of a single αRSC and $\alpha\beta RSE$ in relation to α. The ensemble curve show a slow decline in generalization error for larger pruning values. This is an indication of the robustness of the ensemble in relation to over-pruning when compared to a single classifier. From these experiments we made the following observations concerning the evolution of the ensemble classification accuracy in relation to α:

1. The average accuracy of individual classifiers is significantly lower than ensemble accuracy.
2. Pruning works in the same way for the ensemble as in single classifiers with lesser sharp influence for large values of α.

It is clear from the graphs of Figure 5.2 that we require an accurate classifier in order to use $\alpha\beta RSE$ since a weak classifier is the one that is severely pruned. It seems that severely pruned covers under-represent regions from the training sample which may cause the ensemble to wrongly estimate the decision boundary.

5.3.2.2 β evolution in $\alpha\beta RSE$

In this section, we investigate the evolution of β curves in $\alpha\beta RSE$. Here, β is looked at from a "global" perspective, as it would be infeasible to evaluate each sphere separately. We use the 25 base classifiers to evaluate β and α. Therefore, we call it "total average" since it is based on the totality of all examples for each base classifier averaged over 25 base classifiers (we are not showing the standard deviation because the difference between classifiers are negligible). It is not clear how β values are chosen by the ensemble since the procedure is automatically completed by the sampling process. However, the intuition behind the evolution of β curves in relation to α should be descending. Recall from chapter 3, the characteristics of a Sphere Cover is described as follows:

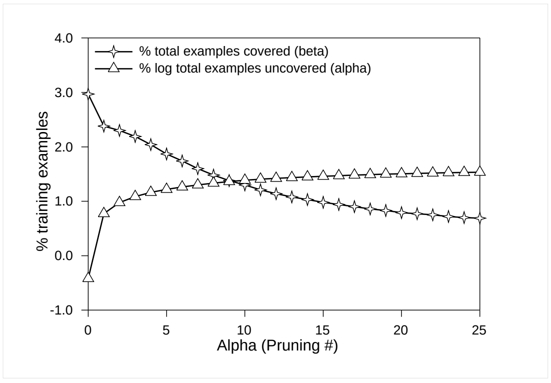
- Spheres that cover large number of examples have centres selected from examples far away from the decision boundary.
- Low cardinality spheres are either close to the decision boundary or they are noise within a dense area of different class label.

Therefore, the number of examples covered from the opposite class depends on the number of spheres in the cover. That is, fewer spheres means covering fewer examples of a different target class. In this case, we will notice more errors that are caused by α as fewer examples are covered. The β parameter plays a further regularization role in regards to noisy examples uncovered by α parameter. Searching for the best values for both parameters is the important step to achieve optimal classification accuracy.

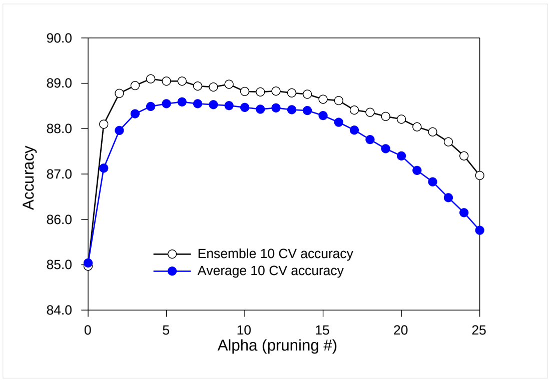
The graphs of Figure 5.2 and 5.3 show the total average percentage of examples covered using β, and the total average percentage of examples uncovered using α[1]. The total averages are based on 25 classifiers per ensemble using the same experimental setup above. With the exception of graph (e) in Figure 5.2 (Pendigitis dataset), we notice downward curves in relation to α parameter indicating pruning interferes directly with β estimates. We believe the exception shown on Pendigitis dataset relates to the distribution of the ten different classes. Therefore, increasing α in Pendigitis uncovers more examples which are then covered by spheres of opposite class. This observation might indicate that Pendigitis has large overlap between the ten classes. In these same graphs, we notice, for $\alpha = 1$, a small percentage of uncovered examples. This may be surprising since we know that a pure and proper cover means the entire training set is covered. However, the covers in the ensemble are altered (perturbed) after each run which eventually uncovers some examples. Furthermore, we notice the two curves in Figure 5.2 evolves in opposite directions (with the exception of curves in graph (e) of Figure 5.2). Indeed, this is an indication that, for small α values, spheres that are removed are the ones that are close to the decision boundary. Uncovered examples are then used in the sampling process to select values for β.

Lastly, we showed that examples with high likelihood of being noise and outliers are being most often involved in the $\alpha\beta RSE$ sampling process. We conclude that in order to diversify the ensemble, we may use the optimal pruning values of a single classifier, then use the proposed sampling in order to automatically estimate values for β. Although, both of these parameters are used for regularization, we hypothesized that β might have a further role in diversifying

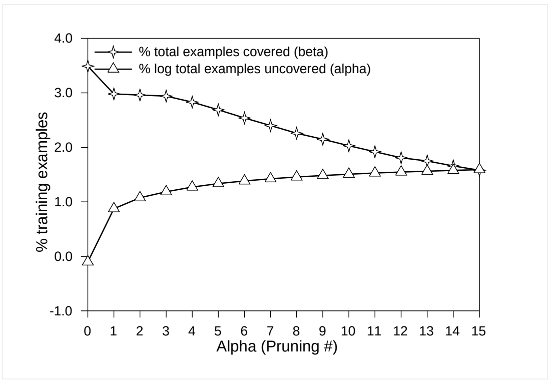
[1]Note, uncovered examples are on a logarithmic scale for better visualization



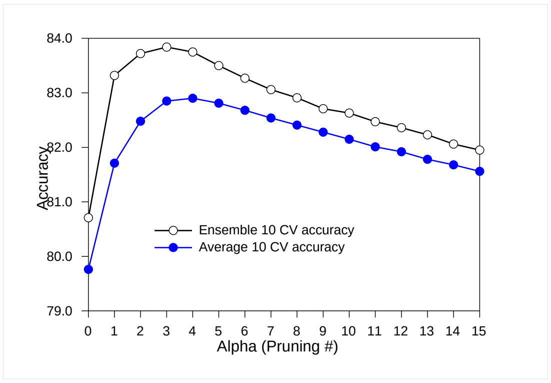
(a) Covered and uncovered curves as a function of α on Clouds dataset



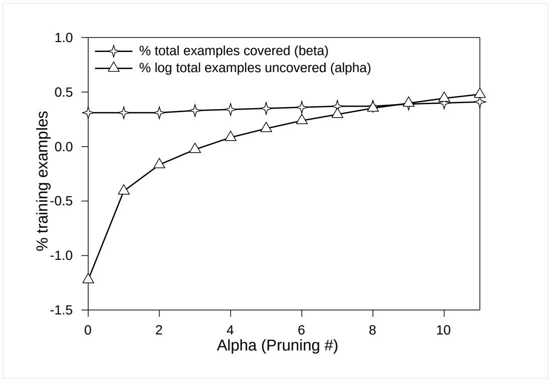
(b) Accuracy as a function of α on Clouds dataset



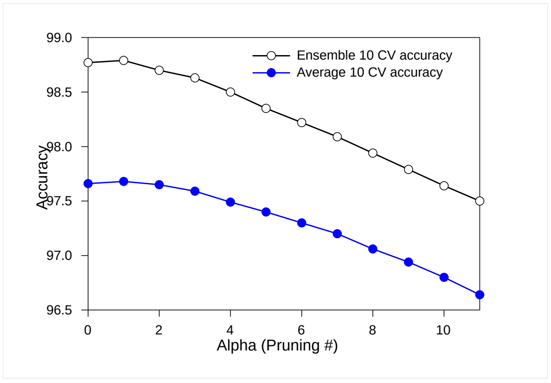
(c) Covered and uncovered curves as a function of α on Magic dataset



(d) Accuracy as a function of α on Magic dataset

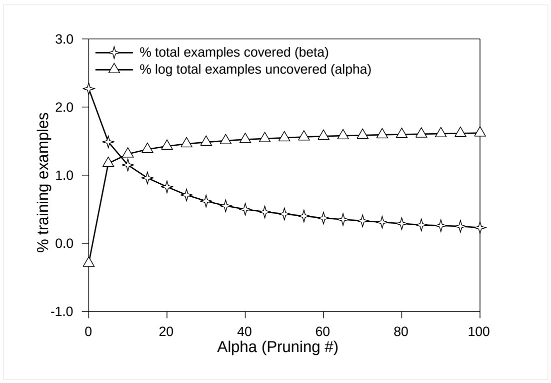


(e) Covered and uncovered curves as a function of α on Pendigitis dataset

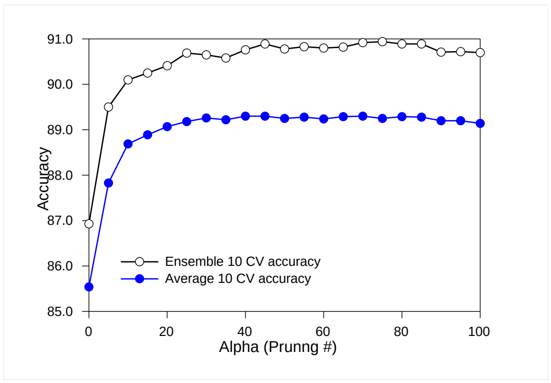


(f) Accuracy as a function of α on Pendigitis dataset

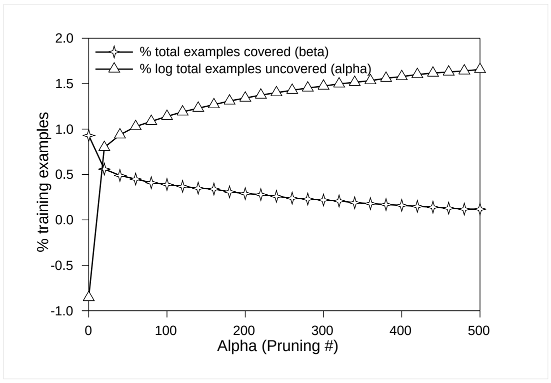
Figure 5.2: Evolution of covered and uncovered curves and, ensemble and averaged classification accuracies in $\alpha\beta RSE$.



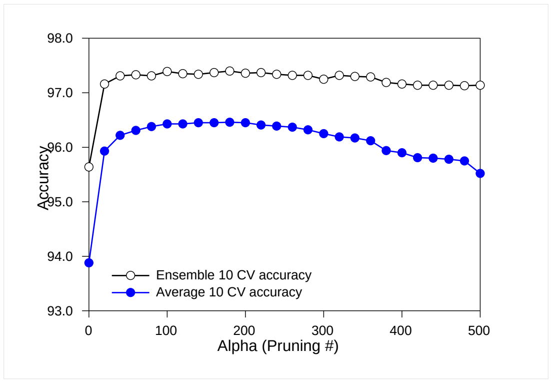
(a) Covered and uncovered curves as a function of α on Waveform dataset



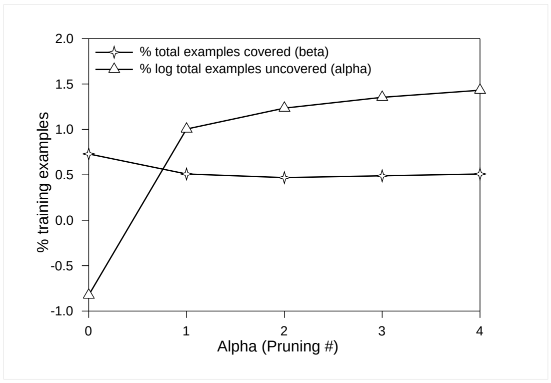
(b) Accuracy as a function of α on Waveform dataset



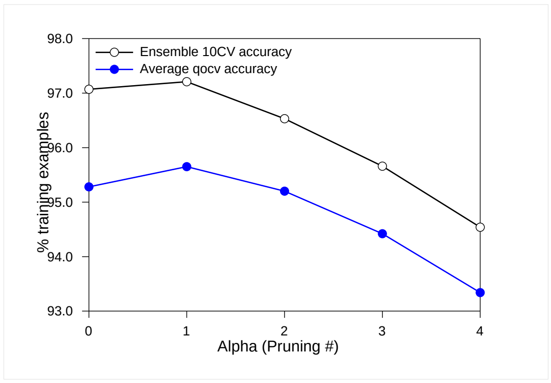
(c) Covered and uncovered curves as a function of α on Twonorm dataset



(d) Accuracy as a function of α on Twonorm dataset



(e) Covered and uncovered curves as a function of α on Ringnorm dataset



(f) Accuracy as a function of α on Ringnorm dataset

Figure 5.3: Evolution of covered and uncovered curves and, ensemble and averaged classification accuracies in $\alpha\beta RSE$.

the ensemble in order to improve on the αRSE algorithm. This is indeed the case as shown by comparing $\alpha\beta RSE$ results with those of αRSE in Table 5.1.

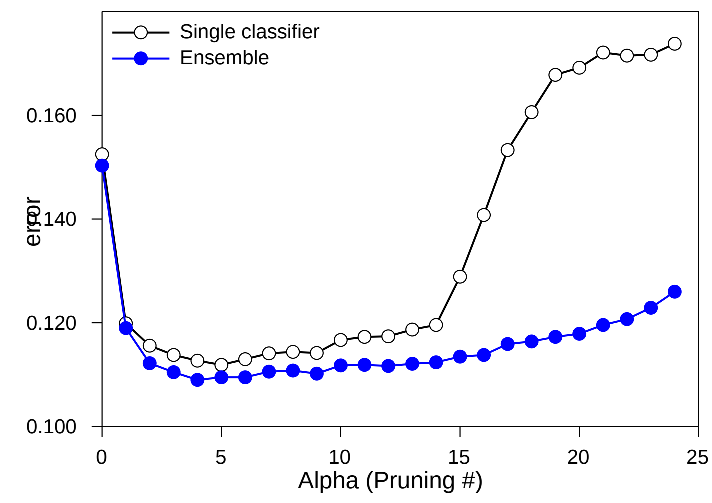


Figure 5.4: Slow decline of generalization error of $\alpha\beta RSE$ on Cloud dataset

5.3.2.3 Bias and Variance Decomposition of $\alpha\beta RSE$

For these experiments, we employ the same datasets used in chapter 3. In order to show the ensemble on bias. We also add Image to make it 11 different datasets. We refer the reader to Table 3.4, in chapter 3, which summarizes the main features of the datasets used here.

We showed, in the bias and variance results of chapter 3, that pruning reduced the unbiased variance of αRSC classifier which resulted in overall loss reduction. We also showed that only in a few cases that pruning reduced bias. However, we noticed that the influence pruning parameter has on bias reduction is weak. Here, we continue with the bias and variance decomposition in order to answer the two questions we asked in the introduction part of this chapter. That is, which is the most affected by the ensemble, bias or variance? Looking at the results in Table 5.4, we notice that $\alpha\beta RSE$ reduces the net variance on all the datasets. However, we notice a very small increase in bias for only 4 datasets. A significant decrease in bias is shown on 3 datasets and very small decrease in bias on 5 datasets. From these results, we can produce three groups that are separated according to their bias and variance results.

Furthermore, these groups will give us further insight into the generalization decomposition of the ensemble.

1. The first group is made of Diabetes, Cloud, and Magic datasets as we notice only a small decrease in bias in comparison to a single classifier.
2. The second group is made of Heart, Pendigitis, Twonorm, Image and Waveform datasets where we observe a small increase in bias in comparison to a single classifier.
3. The third group is made of wdbc, Ringnorm, and Concentric datasets where we see a significant decrease in bias in comparison to a single classifier.

Recall, Bias and variance decomposition tells us that decreasing either the bias or unbiased variance decreases the overall error. However, decreasing biased variance increases the overall error. The summary bias and variance results are shown in Table 5.4. For the first group, we notice that the average error of the ensemble is decreased in comparison with a single classifier for the reason that unbiased variance is decreased. We also notice that the bias of the ensemble has marginally decreased in comparison with a single classifier while biased variance shows significant decrease. Therefore, the significant decrease in net variance in the ensemble caused the decrease in average error.

The second group shows slight decrease in bias for the ensemble in comparison with a single classifier. However, we notice a significant decrease in average error which is caused by a very significant decrease in net variance. Unbiased variance is significantly decreased in the ensemble in comparison with a single classifier whilst biased variance show a smaller but significant increase. The exception is shown for Image dataset (3.11%), and very small increase for Pendigitis dataset (0.94%). The overall average error decreases for the same reason as the group one. That is, the net variance is significantly decreased because the unbiased variance is also decreased.

The third group shows substantial decrease in bias and net variance for the ensemble in comparison with a single classifier. However, wdbc shows small decrease in net variance for the ensemble (5.26%) which is proportional to the decrease of unbiased (7.27%) and biased variance (8.69%). This decrease also explains the average error reduction for the ensemble. In addition, the significant decrease in both bias and net variance is the reason both Ringnorm and Concentric datasets show significant decrease in average error.

We summarize the bias/variance decomposition of the above experiments as we notice a pattern of behaviour for the datasets used:

Table 5.4: Comparing best results of $\alpha\beta RSE$ and αRSC on various datasets using bias and variance decomposition. (Var. unb.) and (Var. bias.) stand for unbiased and biased variance. (Diff) stands for the percentage difference between the two algorithms. The up arrow ↑ means an increase while a down arrow ↓ means a decrease.

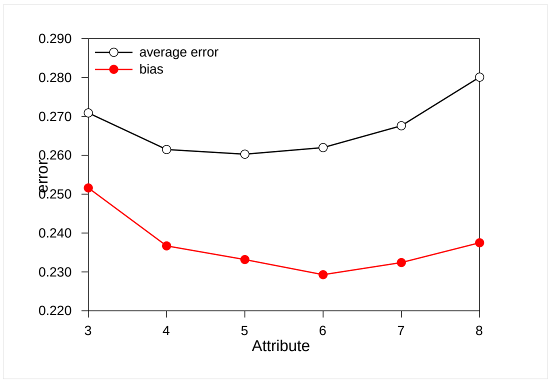
Dataset / Algorithms	*Avg Error*	*Bias*	*Net Var*	*Var. Unb.*	*Var. bias.*
Diabetes					
(1)$\alpha\beta RSE$ $\alpha = 3$	*0.2685*	*0.2359*	*0.0326*	*0.0847*	*0.0521*
(2)αRSC $\alpha = 3$	*0.2780*	*0.2367*	*0.0413*	*0.1006*	*0.0594*
Diff (1) vs (2) %	↓ ***3.41***	↓ ***0.33***	↓ ***21.06***	↓ ***15.80***	↓ ***12.29***
Clouds					
(1)$\alpha\beta RSE$ $\alpha = 3$	*0.1297*	*0.1186*	*0.0111*	*0.0320*	*0.0209*
(2)αRSC $\alpha = 3$	*0.1354*	*0.1196*	*0.0158*	*0.0397*	*0.0240*
Diff (1) vs (2) %	↓ ***4.20***	↓ ***0.83***	↓ ***29.74***	↓ ***19.39***	↓ ***12.91***
Magic					
(1)$\alpha\beta RSE$ $\alpha = 4$	*0.2039*	*0.1900*	*0.0139*	*0.0417*	*0.0277*
(2)αRSC $\alpha = 4$	*0.2151*	*0.1937*	*0.0215*	*0.0556*	*0.0341*
Diff (1) vs (2) %	↓ ***5.20***	↓ ***1.91***	↓ ***35.34***	↓ ***25.00***	↓ ***18.76***
Image					
(1)$\alpha\beta RSE$ $\alpha = 0$	*0.1050*	*0.0665*	*0.0385*	*0.0603*	*0.0218*
(2)αRSC $\alpha = 0$	*0.1184*	*0.0650*	*0.0534*	*0.0759*	*0.0225*
Diff (1) vs (2) %	↓ ***11.31***	↑ ***2.30***	↓ ***27.90***	↓ ***20.55***	↓ ***3.11***
Pendigitis					
(1)$\alpha\beta RSE$ $\alpha = 0$	*0.0958*	*0.0369*	*0.0589*	*0.0697*	*0.0107*
(2)αRSC $\alpha = 0$	*0.1206*	*0.0355*	*0.0850*	*0.0956*	*0.0106*
Diff (1) vs (2) %	↓ ***20.56***	↑ ***3.94***	↓ ***30.70***	↓ ***27.09***	↑ ***0.94***
Twonorm					
(1)$\alpha\beta RSE$ $\alpha = 10$	*0.0345*	*0.0224*	*0.0121*	*0.0179*	*0.0058*
(2)αRSC $\alpha = 10$	*0.0515*	*0.0222*	*0.0293*	*0.0366*	*0.0073*
Diff (1) vs (2)%	↓ ***33.01***	↑ ***0.90***	↓ ***58.70***	↓ ***51.09***	↓ ***20.54***
Waveform					
(1)$\alpha\beta RSE$ $\alpha = 10$	*0.1223*	*0.0976*	*0.0247*	*0.0500*	*0.0254*
(2)αRSC $\alpha = 11$	*0.1387*	*0.0961*	*0.0426*	*0.0722*	*0.0296*
Diff (1) vs (2) %	↓ ***11.82***	↑ ***1.56***	↓ ***42.01***	↓ ***30.74***	↓ ***14.18***
Heart					
(1)$\alpha\beta RSE$ $\alpha = 10$	*0.1896*	*0.1756*	*0.0140*	*0.0431*	*0.0290*
(2)αRSC $\alpha = 7$	*0.2138*	*0.1667*	*0.0471*	*0.0872*	*0.0400*
Diff (1) vs (2) %	↓ ***11.31***	↑ ***5.33***	↓ ***70.27***	↓ ***50.57***	↓ ***27.5***
wdbc					
(1)$\alpha\beta RSE$ $\alpha = 2$	*0.0771*	*0.0663*	*0.0108*	*0.0255*	*0.0147*
(2)αRSC $\alpha = 8$	*0.0898*	*0.0784*	*0.0114*	*0.0275*	*0.0161*
Diff (1) vs (2) %	↓ ***14.14***	↓ ***15.43***	↓ ***5.26***	↓ ***7.27***	↓ ***8.69***
Ringnorm					
(1)$\alpha\beta RSE$ $\alpha = 0$	*0.0527*	*0.0208*	*0.0320*	*0.0377*	*0.0058*
(2)αRSC $\alpha = 0$	*0.1183*	*0.0596*	*0.0587*	*0.0783*	*0.0196*
Diff (1) vs (2) %	↓ ***55.45***	↓ ***65.10***	↓ ***45.48***	↓ ***51.85***	↓ ***70.40***
Concentric					
(1)$\alpha\beta RSE$ $\alpha = 0$	*0.0346*	*0.0121*	*0.0225*	*0.0275*	*0.0049*
(2)αRSC $\alpha = 0$	*0.0616*	*0.0131*	*0.0485*	*0.0544*	*0.0059*
Diff (1) vs (2) %	↓ ***43.83***	↓ ***7.63***	↓ ***53.60***	↓ ***49.44***	↓ ***16.94***

1. The ensemble shows better average error in comparison to a single classifier because of a substantial decrease in net variance. However, there is a slight increase in bias.
2. The ensemble shows better average error in comparison to a single classifier because of a significant decrease in bias.
3. The ensemble shows better average error in comparison to a single classifier because of a significant decrease in both bias and net variance.

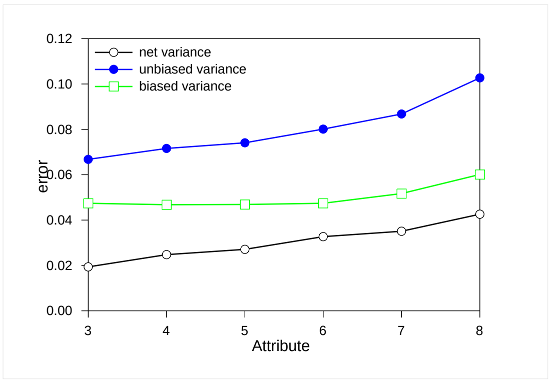
We conclude from the above results that $\alpha\beta RSE$, in most cases, reduces the net variance in comparison with a single classifier because of a decrease in unbiased variance. However, it is not straightforward in relation to bias. It might be that bias reduction depends on the geometrical complexity of the sample (complex structures require complex decision boundaries), the chosen values for the pruning parameter α, and the interaction between α and β (as shown in section 5.3.2.2). In that case, finding a method that systematically reduces bias while keeping unbiased variance low will further reduce the ensemble average error.

5.3.2.4 Bias and Variance Decomposition of $\alpha RSSE$

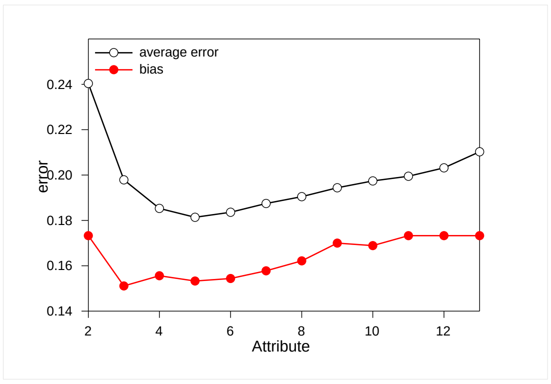
In this section, we show the results of the bias and variance decomposition error of $\alpha RSSE$. We followed the same experimental format in the previous section in order to make direct comparisons. The curves showing both bias and variance in relation to κ is depicted in Figures 5.5. We notice a strong relationship between averaged error and bias. This is shown on all the datasets with the exception of Twonorm dataset where bias does not show the same strong decrease. This first observation is an indication that $\alpha RSSE$ reduces bias by increasing κ. We notice from the same graphs that each time κ increases, net variance decreases in a significant way with the exception of Diabetes. For this dataset, average error is decreased solely by the decrease in bias, since both unbiased and biased variance increase with increasing κ. For Yeast dataset, we notice an inexplicable increase of unbiased variance for $\kappa = 3$ then a decrease up to $\kappa = 5$ followed by an increase again. The same trend is noticed for net variance since biased variance shows a continuous decrease. For the remaining datasets, unbiased variance decrease in "U" shape form whilst this is not the case for biased variance. Increasing κ seems to have a higher influence on unbiased variance reduction than biased variance. However, to be able to find out whether there is bias decrease in comparison with αRSC and αRSE, we need to make specific assessment as shown in Table 5.5.



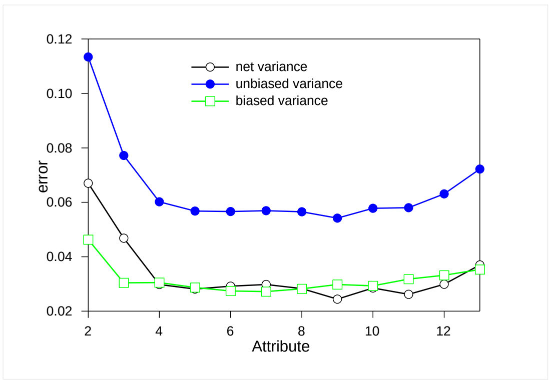
(a) $\alpha RSSE$ Average error and bias decomposition of the Diabetes dataset



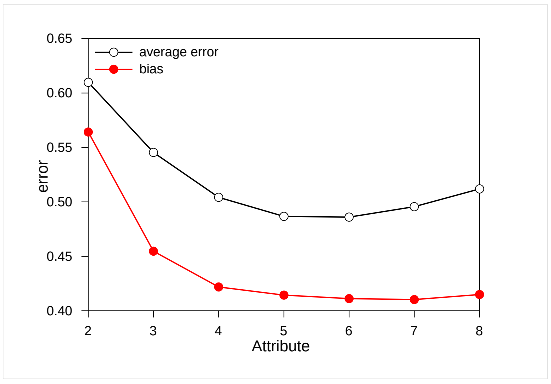
(b) $\alpha RSSE$ Variances decomposition of the diabetes dataset



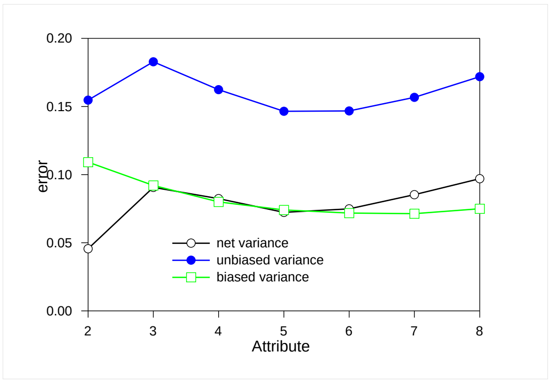
(c) $\alpha RSSE$ Average error and bias decomposition of the Heart dataset



(d) $\alpha RSSE$ Variances decomposition of the heart dataset

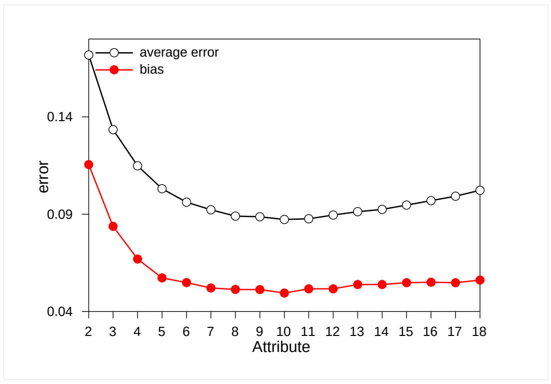


(e) $\alpha RSSE$ Average error and bias decomposition of the Yeast dataset

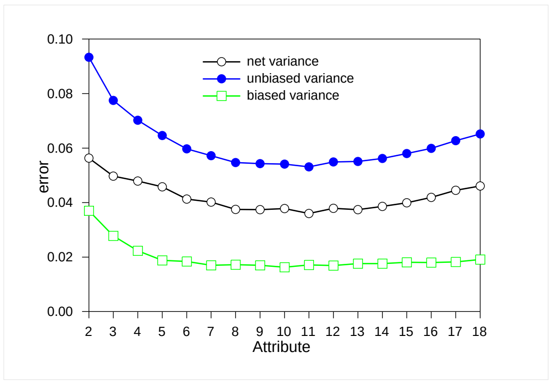


(f) $\alpha RSSE$ Variances decomposition of the Yeast dataset

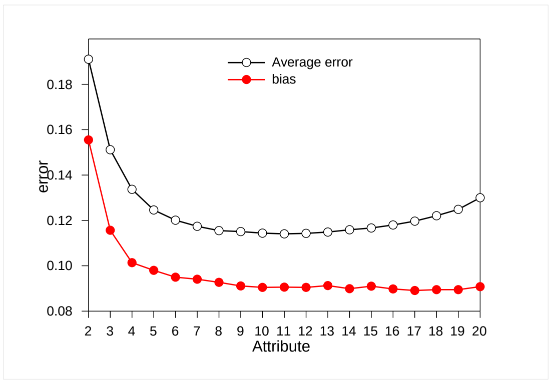
Figure 5.5: The Bias/Variance Decomposition of the $\alpha RSSE$ classifier



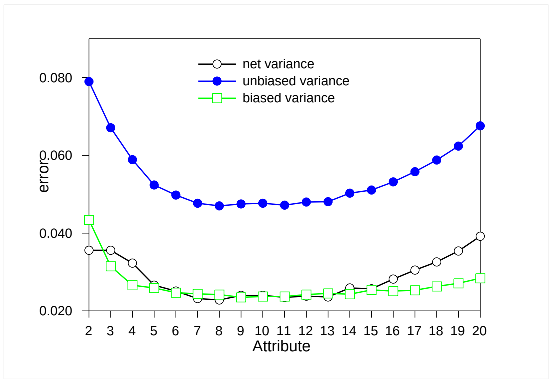
(a) $\alpha RSSE$ Average error and bias decomposition of the Image dataset



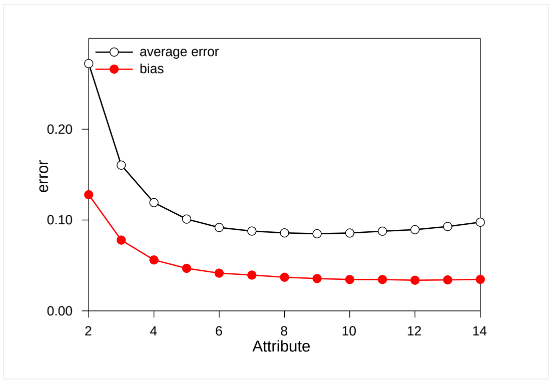
(b) $\alpha RSSE$ Variances decomposition of the Image dataset



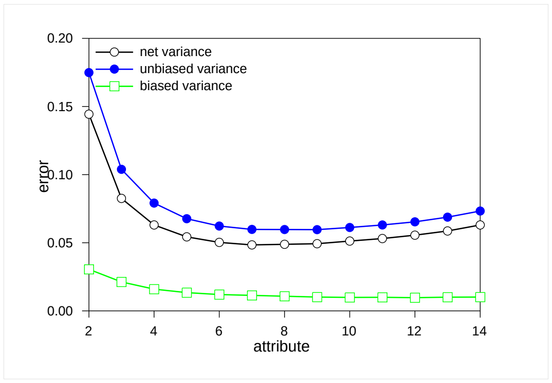
(c) $\alpha RSSE$ Average error and bias decomposition of the Waveform dataset



(d) $\alpha RSSE$ Variances decomposition of the Waveform dataset



(e) $\alpha RSSE$ Average error and bias decomposition of the Pendigitis dataset



(f) $\alpha RSSE$ Variances decomposition of the Pendigitis dataset

Figure 5.6: The Bias/Variance Decomposition of the $\alpha RSSE$ classifier

Table 5.5 shows the bias/variance decomposition of $\alpha RSSE$, $\alpha\beta RSE$ and αRSC. A closer look at the values in Table 5.5 shows decrease in bias on all the datasets with the exception of Twonorm dataset where the bias has slightly increased. We also notice a slight increase on Pendigitis in comparison with $\alpha RSSE$ and αRSC. Both results are insignificant in comparison to the overwhelming decrease in bias noticed on all the remaining datasets. This is clearly an indication that $\alpha RSSE$ reduce bias. As for $\alpha RSSE$ variance, we notice substantial decrease in unbiased variance with the exception of Heart dataset where it shows an increase. A slight increase is also shown on wdbc dataset. We can conclude that the overall average errors decrease for the reason that both bias and unbiased variances are decreased. This explains the performance of $\alpha RSSE$ in comparison with αRSC and $\alpha\beta RSE$.

5.3.3 Experiment 3: Comparing Classifiers Accuracy Results

5.3.3.1 Comparing αRSE and $\alpha\beta RSE$ Accuracies against Other Ensembles

Table 5.6 shows the classification accuracy of αRSE and $\alpha RSSE$ accuracies against those of Adaboost, Bagging, and Multiboost trained on 25 base classifiers.
Friedman ranks $\alpha\beta RSE$ and αRSE 1^{th} and 5^{th} respectively, whilst AdaBoost and Bagging are ranked 3^{th} and 4^{th} respectively. Multiboost ranked 2^{nd}. These results demonstrate that $\alpha\beta RSE$ performance is better than AdaBoost and Bagging using 25 base classifiers. αRSE results are similar to those of AdaBoost and Bagging which explains the similarity of average ranks.

In general, ensembles perform better when the size of the ensemble is large. Table 5.7 shows the classification performance for ensemble size based on 100 base classifiers. $\alpha\beta RSE$ ranked 1^{th} whilst αRSE lost it 3^{rd} position to Adaboost. The average ranks show that αRSE performed similarly to Bagging, and $\alpha\beta RSE$ to Multiboost. These experiments indicate that $\alpha\beta RSE$ performs well for these datasets.

5.3.3.2 Comparing $\alpha RSSE$ Accuracy against Other Subspace Ensembles

Table 5.8 shows the classification accuracy of $\alpha RSSE$ accuracy against those of Rotation Forest, Random SubSpace, Random Forest, and Random Committee based on 25 classifiers.

Friedman ranks $\alpha RSSE$ 2^{nd}, whilst Rotation Forest is ranked 1^{st}. However, the average ranks of both algorithms are very similar. Random Subspaces ranked last whilst Random Forest and Random Committee are ranked 3^{rd} and 4^{th} respectively. These results demonstrate

Table 5.5: Comparing Bias/variance of $\alpha RSSE$,$\alpha\beta RSE$ and αRSC on various datasets using bias and variance. (Var. unb.) and (Var. bias.) stand for unbiased and biased variance. (Diff) stands for the percentage difference between the algorithms. The up arrow $\uparrow$ means an increase while a down arrow $\downarrow$ means a decrease.

Dataset	*Avg Error*	*Bias*	*Net Var*	*Var. Unb.*	*Var. bias.*
Diabetes					
$(1)\alpha RSSE$, $\alpha = 2$, $\kappa = 5$	*0.2603*	*0.2332*	*0.0271*	*0.0741*	*0.0469*
$(2)\alpha\beta RSE$, $\alpha = 3$	*0.2685*	*0.2359*	*0.0326*	*0.0847*	*0.0521*
$(3)\alpha RSC$, $\alpha = 3$	*0.2780*	*0.2367*	*0.0413*	*0.1006*	*0.0594*
Diff (1) vs (2) %	$\downarrow$ ***3.05***	$\downarrow$ ***1.14***	$\downarrow$ ***16.87***	$\downarrow$ ***12.51***	$\downarrow$ ***9.98***
Diff (1) vs (3) %	$\downarrow$ ***6.37***	$\downarrow$ ***1.48***	$\downarrow$ ***34.38***	$\downarrow$ ***26.34***	$\downarrow$ ***21.04***
Heart					
$(1)\alpha RSSE$, $\alpha = 2$, $\kappa = 5$	*0.1814*	*0.1533*	*0.0281*	*0.0568*	*0.0287*
$(2)\alpha\beta RSE$, $\alpha = 10$	*0.1896*	*0.1756*	*0.0140*	*0.0431*	*0.0290*
$(3)\alpha RSC$, $\alpha = 7$	*0.2138*	*0.1667*	*0.0471*	*0.0872*	*0.0400*
Diff (1) vs (2) %	$\downarrow$ ***4.32***	$\downarrow$ ***12.70***	$\uparrow$ ***100.71***	$\uparrow$ ***31.79***	$\downarrow$ ***1.034***
Diff (1) vs (3) %	$\downarrow$ ***15.15***	$\downarrow$ ***8.04***	$\downarrow$ ***40.34***	$\downarrow$ ***34.86***	$\downarrow$ ***28.25***
wdbc					
$(1)\alpha RSSE$, $\alpha = 0$, $\kappa = 13$	*0.0698*	*0.0553*	*0.0145*	*0.0258*	*0.0112*
$(2)\alpha\beta RSE$, $\alpha = 2$	*0.0771*	*0.0663*	*0.0108*	*0.0255*	*0.0147*
$(3)\alpha RSC$, $\alpha = 8$	*0.0898*	*0.0784*	*0.0114*	*0.0275*	*0.0161*
Diff (1) vs (2) %	$\downarrow$ ***9.46***	$\downarrow$ ***16.59***	$\uparrow$ ***34.25***	$\uparrow$ ***1.17***	$\downarrow$ ***23.80***
Diff (1) vs (3) %	$\downarrow$ ***22.27***	$\downarrow$ ***29.46***	$\uparrow$ ***27.19***	$\downarrow$ ***6.18***	$\downarrow$ ***30.43***
Image					
$(1)\alpha RSSE$, $\alpha = 0$, $\kappa = 10$	*0.0873*	*0.0495*	*0.0378*	*0.0541*	*0.0163*
$(2)\alpha\beta RSE$, $\alpha = 0$	*0.1050*	*0.0665*	*0.0385*	*0.0603*	*0.0218*
$(3)\alpha RSC$, $\alpha = 0$	*0.1184*	*0.0650*	*0.0534*	*0.0759*	*0.0225*
Diff (1) vs (2) %	$\downarrow$ ***16.85***	$\downarrow$ ***25.56***	$\downarrow$ ***1.81***	$\downarrow$ ***10.28***	$\downarrow$ ***25.22***
Diff (1) vs (3) %	$\downarrow$ ***26.26***	$\downarrow$ ***23.84***	$\downarrow$ ***29.21***	$\downarrow$ ***28.72***	$\downarrow$ ***27.55***
Pendigitis					
$(1)\alpha RSSE$, $\alpha = 0$, $\kappa = 9$	*0.0849*	*0.0356*	*0.0493*	*0.0596*	*0.0102*
$(2)\alpha\beta RSE$, $\alpha = 0$	*0.0958*	*0.0369*	*0.0589*	*0.0697*	*0.0107*
$(3)\alpha RSC$, $\alpha = 0$	*0.1206*	*0.0355*	*0.0850*	*0.0956*	*0.0106*
Diff (1) vs (2) %	$\downarrow$ ***11.37***	$\downarrow$ ***3.52***	$\downarrow$ ***16.29***	$\downarrow$ ***14.49***	$\downarrow$ ***4.67***
Diff (1) vs (3) %	$\downarrow$ ***29.60***	$\uparrow$ ***0.28***	$\downarrow$ ***42.00***	$\downarrow$ ***37.65***	$\downarrow$ ***3.77***
Twonorm					
$(1)\alpha RSSE$, $\alpha = 2$, $\kappa = 13$	*0.0328*	*0.0225*	*0.0103*	*0.0159*	*0.0057*
$(2)\alpha\beta RSE$, $\alpha = 10$	*0.0345*	*0.0224*	*0.0121*	*0.0179*	*0.0058*
$(3)\alpha RSC$, $\alpha = 10$	*0.0515*	*0.0222*	*0.0293*	*0.0366*	*0.0073*
Diff (1) vs (2)%	$\downarrow$ ***4.92***	$\uparrow$ ***0.44***	$\downarrow$ ***14.87***	$\downarrow$ ***11.17***	$\downarrow$ ***1.72***
Diff (1) vs (3)%	$\downarrow$ ***36.31***	$\uparrow$ ***1.35***	$\downarrow$ ***64.84***	$\downarrow$ ***56.55***	$\downarrow$ ***21.91***
Waveform					
$(1)\alpha RSSE$, $\alpha = 2$, $\kappa = 11$	*0.1141*	*0.0906*	*0.0235*	*0.0472*	*0.0237*
$(2)\alpha\beta RSE$, $\alpha = 10$	*0.1223*	*0.0976*	*0.0247*	*0.0500*	*0.0254*
$(3)\alpha RSC$, $\alpha = 11$	*0.1387*	*0.0961*	*0.0426*	*0.0722*	*0.0296*
Diff (1) vs (2) %	$\downarrow$ ***6.70***	$\downarrow$ ***7.17***	$\downarrow$ ***4.85***	$\downarrow$ ***5.60***	$\downarrow$ ***6.69***
Diff (1) vs (3) %	$\downarrow$ ***17.73***	$\downarrow$ ***5.72***	$\downarrow$ ***44.83***	$\downarrow$ ***34.62***	$\downarrow$ ***19.93***
Ringnorm					
$(1)\alpha RSSE$ $\alpha = 0$, $\kappa = 10$	*0.0288*	*0.0167*	*0.0121*	*0.0166*	*0.0045*
$(2)\alpha\beta RSE$, $\alpha = 0$	*0.0527*	*0.0208*	*0.032*	*0.0377*	*0.0058*
$(3)\alpha RSC$, $\alpha = 0$	*0.1183*	*0.0596*	*0.0587*	*0.0783*	*0.0783*
Diff (1) vs (2) %	$\downarrow$ ***45.35***	$\downarrow$ ***19.71***	$\downarrow$ ***62.18***	$\downarrow$ ***55.96***	$\downarrow$ ***22.41***
Diff (1) vs (3) %	$\downarrow$ ***75.65***	$\downarrow$ ***71.97***	$\downarrow$ ***79.38***	$\downarrow$ ***78.79***	$\downarrow$ ***94.25***

Table 5.6: Classification accuracy (in %) and standard deviation of $\alpha\beta RSE$, αRSE, $\alpha RSSE$, Adaboost, Bagging, and Multiboost using average results of 30 different runs on independent train/test splits. The ensembles use 25 base classifier. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Data Set	αRSE	$\alpha\beta RSE$	Adaboost	Bagging	MultiBoost
Abalone	54.25 ± 0.94	**54.89** ± 1.02	52.30 ± 1.20	53.98 ± 0.91	53.04 ± 1.47
waveform	90.40 ± 0.67	**90.68** ± 0.65	89.60 ± 0.69	88.71 ± 0.58	89.63 ± 0.56
satimage	90.90 ± 0.41	90.90 ± 0.41	**91.21** ± 0.45	89.82 ± 0.69	90.94 ± 0.57
ringnorm	96.71 ± 0.38	97.17 ± 0.30	**97.26** ± 0.33	95.01 ± 0.50	97.12 ± 0.31
twonorm	97.32 ± 0.26	**97.41** ± 0.26	96.43 ± 0.32	95.58 ± 0.46	96.41 ± 0.37
image	96.87 ± 0.50	96.87 ± 0.51	**97.77** ± 0.64	95.78 ± 0.90	97.32 ± 0.75
german	73.21 ± 1.76	74.00 ± 1.69	74.52 ± 1.76	**75.24** ± 1.36	75.09 ± 2.51
wdbc	93.21 ± 1.47	93.86 ± 1.52	**96.79** ± 1.26	95.19 ± 1.38	96.61 ± 1.22
yeast	56.34 ± 2.09	58.22 ± 1.24	58.23 ± 1.59	**60.65** ± 1.57	58.65 ± 1.77
diabetes	74.52 ± 1.78	75.01 ± 1.79	73.54 ± 1.88	**75.94** ± 2.00	74.74 ± 2.34
iono	**93.48** ± 2.05	93.39 ± 2.25	92.85 ± 2.20	92.31 ± 2.60	93.25 ± 2.05
sonar	**84.67** ± 4.17	84.43 ± 3.66	81.38 ± 4.21	76.33 ± 5.66	80.76 ± 4.57
heart	78.85 ± 3.60	80.74 ± 3.26	80.41 ± 3.11	**81.26** ± 3.66	81.22 ± 2.87
cancer	69.46 ± 2.97	70.07 ± 3.62	69.07 ± 4.36	**73.44** ± 2.87	69.35 ± 4.71
winsc	95.53 ± 1.34	95.67 ± 1.33	96.21 ± 0.84	96.01 ± 0.97	**96.49** ± 0.71
ecoli	85.36 ± 2.78	**85.51** ± 2.64	83.07 ± 2.75	83.45 ± 3.58	83.45 ± 2.73
Average Ranks	3.31	2.50	3.13	3.28	2.78
Ranks	5	1	3	4	2

that $\alpha RSSE$ performance is as good as Rotation Forest, and better than state of the art Random subspaces, Random Forest, and Random Committee using 25 base classifiers. This is also shown in Table 5.7 for ensemble size of 100 base classifiers. $\alpha RSSE$ still ranked 2^{nd} and the average ranks show both Rotation Forest and $\alpha RSSE$ performed similarly for ensemble size of 100 base classifiers. These experiments indicate that $\alpha\beta RSE$ performs well for these datasets.

5.3.4 Experiment 5: On the Performance of Various $\alpha RSSE$ Sizes

Table 5.10 shows the classification accuracy of $\alpha RSSE$ for various sizes varying from 15 to 500 base classifiers using 10CV. In general, ensembles perform much better when the size of the ensemble is large. However, over training often degrades performance for many ensemble methods. It is interesting to note that $\alpha RSSE$ improves its classification performance for ensemble size that are above 100 base classifiers. We notice from Table 5.10, 8 improvements are made for ensembles made of 500 base classifiers and 7 for ensembles made of 250 base classifiers.

Table 5.7: Classification accuracy (in %) and standard deviation of $\alpha\beta RSE$, αRSE, $\alpha RSSE$, Adaboost, Bagging, and Multiboost using average results of 30 different runs on independent train/test splits. The ensembles use 100 base classifier. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Data Set	αRSE	$\alpha\beta RSE$	Adaboost	Bagging	MultiBoost
Abalone	54.36 ± 1.16	**54.48** ± 1.23	52.82 ± 0.99	54.1 ± 0.91	54.22 ± 1.47
waveform	**90.56** ± 0.70	90.32 ± 0.66	90.27 ± 0.58	89.08 ± 0.84	90.20 ± 0.93
satimage	90.91 ± 0.38	91.12 ± 0.44	**92.00** ± 0.39	90.47 ± 0.55	91.11 ± 0.60
ringnorm	96.88 ± 0.37	97.54 ± 0.31	**97.75** ± 0.29	95.23 ± 0.52	97.05 ± 0.52
twonorm	97.36 ± 0.28	**97.49** ± 0.22	97.13 ± 0.26	96.35 ± 0.38	96.95 ± 0.27
image	96.77 ± 0.50	96.8 ± 0.56	**97.98** ± 0.56	96.23 ± 0.80	96.71 ± 0.34
german	73.23 ± 1.82	74.16 ± 1.58	74.46 ± 1.54	**74.91** ± 1.85	74.70 ± 0.64
wdbc	93.39 ± 1.56	93.91 ± 1.57	**96.91** ± 1.55	96.33 ± 1.35	96.47 ± 1.07
yeast	57.26 ± 1.44	58.41 ± 1.36	58.13 ± 1.62	**60.08** ± 1.56	59.57 ± 1.22
diabetes	74.53 ± 1.84	75.04 ± 2.57	73.53 ± 2.20	**75.68** ± 2.57	74.54 ± 1.28
iono	**93.56** ± 2.06	93.53 ± 1.96	92.99 ± 2.29	91.20 ± 3.01	92.39 ± 2.25
sonar	84.86 ± 4.23	**85.00** ± 3.72	82.71 ± 5.14	78.57 ± 5.86	82.71 ± 2.21
heart	79.26 ± 3.40	80.67 ± 3.10	81.19 ± 2.88	81.56 ± 3.59	**82.33** ± 4.20
cancer	69.53 ± 3.29	69.58 ± 3.32	68.82 ± 5.07	**73.19** ± 3.34	71.33 ± 3.51
winsc	95.54 ± 1.33	95.71 ± 1.33	96.48 ± 0.88	96.09 ± 0.94	**97.00** ± 4.31
ecoli	85.54 ± 2.96	**85.86** ± 2.65	83.07 ± 2.75	83.45 ± 3.58	84.82 ± 0.75
Average Ranks	3.38	2.38	3.03	3.44	2.78
Ranks	4	1	3	5	2

Table 5.8: Classification accuracy (in %) and standard deviation of $\alpha RSSE$, Rotation Forest, Random SubSpace, RandomForest and Random Committee using average results of 30 different runs on independent train/test splits. The ensembles use 25 base classifier. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Data Set	$\alpha RSSE$	Rotation Forest	Random SubSpace	Random Forest	Random Committee
Abalone	54.77 ± 1.28	**55.56** ± 1.04	54.62 ± 1.09	54.05 ± 1.16	53.56 ± 1.19
Waveform	90.21 ± 0.51	**90.72** ± 0.77	89.35 ± 0.73	89.51 ± 0.61	89.32 ± 0.61
Satimage	**91.71** ± 0.47	91.03 ± 0.50	90.79 ± 0.54	90.80 ± 0.52	90.24 ± 0.44
Ringnorm	**98.29** ± 0.26	97.57 ± 0.23	96.82 ± 0.35	95.49 ± 0.38	96.6 ± 0.30
Twonorm	97.03 ± 0.30	**97.42** ± 0.27	95.88 ± 0.33	96.02 ± 0.37	96.18 ± 0.35
Image	97.39 ± 0.65	**98.04** ± 0.51	96.42 ± 0.73	97.27 ± 0.63	96.08 ± 0.58
German	74.59 ± 1.47	**76.26** ± 1.63	72.28 ± 1.53	74.85 ± 1.46	73.65 ± 1.77
wdbc	94.67 ± 1.33	**96.40** ± 1.03	95.35 ± 1.31	95.30 ± 1.42	96.04 ± 1.26
Yeast	58.80 ± 1.90	**61.06** ± 1.82	57.38 ± 2.45	58.96 ± 1.69	60.26 ± 1.75
Diabetes	76.17 ± 2.25	**76.25** ± 2.30	74.48 ± 1.98	75.43 ± 1.92	74.78 ± 1.51
Iono	**94.53** ± 1.79	93.50 ± 1.79	92.68 ± 2.40	93.05 ± 1.86	93.13 ± 2.33
Sonar	**84.52** ± 4.49	82.86 ± 4.50	79.57 ± 5.24	81 ± 4.68	82.19 ± 3.99
Heart	**82.74** ± 4.02	**82.74** ± 3.32	83.30 ± 3.55	81.67 ± 3.17	81.00 ± 3.62
Cancer	**76.27** ± 2.96	73.87 ± 3.29	74.73 ± 2.81	71.18 ± 3.74	70.93 ± 4.29
Winsc	**97.21** ± 0.95	97.18 ± 0.83	96.35 ± 1.01	96.48 ± 0.72	97.00 ± 0.84
Ecoli	85.00 ± 2.07	**87.41** ± 2.44	84.02 ± 3.13	85.33 ± 2.76	84.82 ± 2.62
Average Ranks	2.09	1.53	4.00	3.50	3.88
Ranks	2	1	5	3	4

Table 5.9: Classification accuracy (in %) and standard deviation of $\alpha RSSE$, Rotation Forest, Random SubSpace, RandomForest and Random Committee using average results of 30 different runs on independent train/test splits. The ensembles use 100 base classifier. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Data Set	$\alpha RSSE$	Rotation Forest	Random SubSpace	Random Forest	Random Committee
Abalone	54.91 ± 0.98	**56.04** ± 1.04	54.79 ± 1.02	54.47 ± 0.86	52.83 ± 0.95
waveform	90.73 ± 0.53	**91.07** ± 0.77	89.68 ± 0.62	89.97 ± 0.62	90.36 ± 0.63
satimage	**91.92** ± 0.54	91.70 ± 0.50	91.28 ± 0.55	91.59 ± 0.46	91.82 ± 0.46
ringnorm	**98.43** ± 0.27	97.77 ± 0.23	97.22 ± 0.35	95.66 ± 0.43	97.70 ± 0.26
twonorm	97.39 ± 0.28	**97.53** ± 0.27	96.24 ± 0.51	96.38 ± 0.50	97.22 ± 0.27
image	97.83 ± 0.53	**98.16** ± 0.51	96.78 ± 0.62	97.45 ± 0.62	97.93 ± 0.56
german	74.28 ± 1.56	**75.69** ± 1.63	72.37 ± 1.06	75.63 ± 0.64	74.79 ± 1.86
wdbc	95.00 ± 1.44	96.75 ± 1.03	96.35 ± 1.49	96.95 ± 1.17	**97.11** ± 1.32
yeast	59.43 ± 1.93	**61.65** ± 1.82	58.94 ± 1.84	60.03 ± 1.31	58.22 ± 1.57
diabetes	**76.25** ± 2.21	76.12 ± 2.30	74.84 ± 2.07	75.14 ± 2.04	74.00 ± 2.02
iono	**94.76** ± 1.68	94.19 ± 1.79	92.74 ± 1.80	92.39 ± 1.77	93.33 ± 1.94
sonar	**85.24** ± 5.39	84.43 ± 4.50	79.62 ± 5.62	82.05 ± 4.44	82.24 ± 4.63
heart	**84.00** ± 3.43	83.30 ± 3.15	83.41 ± 3.92	82.70 ± 3.35	81.22 ± 4.50
cancer	**76.16** ± 2.75	74.12 ± 3.29	75.30 ± 2.85	71.36 ± 4.41	68.82 ± 5.07
winsc	**97.42** ± 0.91	97.38 ± 0.83	96.60 ± 0.98	96.71 ± 0.90	96.47 ± 0.78
ecoli	85.71 ± 2.36	**87.41** ± 2.44	84.02 ± 3.13	85.33 ± 2.76	83.45 ± 2.73
Average Ranks	1.94	1.69	4.06	3.50	3.81
Ranks	2	1	5	3	4

Table 5.10: $\alpha RSSE$ 10CV accuracy using various ensemble sizes

Dataset/ensemble size	(15)	(25)	(50)	(100)	(250)	(500)
wins	97.08	97.34	97.24	**97.40**	97.38	97.33
Cancer	75.54	75.88	76.05	77.06	76.93	**77.30**
Diabetes	76.89	76.95	76.96	77.03	**77.21**	76.96
German	74.77	75.43	75.52	75.47	75.52	**75.66**
Ecoli	86.17	86.45	86.57	86.15	**86.62**	86.60
Glass2	95.76	**96.76**	96.43	96.33	96.29	96.67
Sonar	86.85	87.81	88.30	**88.69**	88.03	88.47
Iono	95.09	95.37	95.23	95.11	**95.46**	95.43
Glass6	76.96	77.39	77.71	79.08	**79.14**	78.43
Heart	81.74	**84.26**	83.85	83.63	83.81	83.96
wdbc	97.27	97.45	97.75	97.68	**97.99**	97.98
Vowel	98.60	98.79	98.83	98.72	**98.87**	98.85
Yeast	59.02	59.79	59.56	59.58	59.86	**59.94**
Image	97.44	97.80	97.92	97.87	98.01	**98.03**
Pendigits	98.92	98.99	99.03	99.07	99.06	**99.09**
Waveform	89.87	90.38	90.72	90.85	**91.21**	90.97
Magic	84.42	84.89	85.10	85.13	85.28	**85.34**
Twonorm	96.79	97.20	97.39	97.49	97.63	**97.64**
Ringnorm	97.97	98.14	98.27	98.31	98.37	**98.39**

5.4 Chapter Summary

This chapter proposed three ensemble algorithms based on Randomized Sphere Cover classifiers. The main issue we faced, for using a sphere cover algorithm in an ensemble, is the way in which a cover is selected. Searching for the minimum cover will reduce the diversity in the ensemble. Therefore, we select random covers and built an ensemble based on the geometrical property of αRSC classifier. The main idea is to capture the decision boundary via αRSC local spherical edges. We employed two different ways for generating divers covers for both αRSE and $\alpha\beta RSE$. The first method performs a random selection of centres and the second method employs β parameter in order to further randomise the selected covers. In order to keep accurate base classifiers, the sampling process is used on border examples as found by the base classifiers. Consequently, reducing the effect of "complete" random covers which might produce weak classifiers. We showed that $\alpha\beta RSE$ performs well on various datasets in comparison with various well known ensemble methods. Good generalization performance of $\alpha\beta RSE$ is mainly caused by the reduction in net variance. Furthermore, we showed that aggregating αRSC by simple voting can also improve the classification performance over a single classifier. In general, the sampling method used to generate values for β in $\alpha\beta RSE$ improved the overall classification accuracy in comparison with αRSE. We believe that other sampling methods could be devised in order to improve further $\alpha\beta RSE$ performance. The experiments conducted in this chapter indicate that ensembles based on αRSC classifiers performs better in the subspaces. We used bias and variance decomposition on a variety of datasets to investigate the reasons of such improvement. The experiments showed that the decrease of the classification error was mainly due to bias and unbiased variance reduction.

Chapter 6

Application to Gene Expression Classification

6.1 Introduction

Finally, we explore our proposed subspace algorithm $\alpha RSSE$ using various experiments on gene expression datasets. We use three attribute ranking methods on these gene expression datasets in order to verify the usefulness of the sphere cover algorithms proposed in this theses.

Gene expression profiling helps to identify a set of genes that are responsible for cancerous tissue. In the last decade, microarray gene expression cancer diagnosis showed promising results using various classification algorithm. Among those successful algorithms are SVM, and decision forest. In this section we test the performance of αSCC algorithm on seven gene expression datasets.

In supervised learning, the attribute selection problem is defined as: given a set of candidate attributes select a subset defined by one of three approaches [97]:

1. The subset with a specified size that optimizes an evaluation measure.
2. The subset of smaller size that satisfies a certain restriction.
3. The subset with the best commitment among its size and the value of its evaluation measure (general case).

From a supervised learning perspective, the relevance of an attributes with respect to noise reduction and consequently better class separation is the objective that is looked for. Molina et al [97], surveyed and tested different domains on a large repertoire of attribute selection methods exposing in detail their merits and failings. Another exhaustive survey describes the relative difficulty in choosing a specific attribute selection method, hence giving some guidance on how to choose a method specific to a particular domain based on several criteria [51].

Ranking attributes according to a specific statistical evaluation method are popular because of their simplicity, scalability, and good empirical success[97]. Ranking methods showed to perform particularly well For gene expression datasets [85]. Genes, which are represented as attributes, are ranked according to their prediction power and their contribution to class separability. We will use two ranking methods in this section in order to evaluate the proposed classifiers of previous chapters with gene expression datasets. These two popular methods rank best attributes according to the χ^2 statistics and Information Gain (IF). Guyon and Elisseeff [51], outlined important points concerning ranking methods based on evaluating single attributes separately, below we enumerate their main conclusion.

1. Perfectly correlated variables are truly redundant in the sense that no additional information is gained by adding them.

2. Very high variable correlation (or anti-correlation) does not mean absence of variable complementarity.

3. a variable that is completely useless by itself can provide a significant performance improvement when taken with others.

4. Two variables that are useless by themselves can be useful together.

5. A variable useless by itself can be useful together with others.

This is clearly an issue for ranking methods, for that reason many methods that evaluates subsets of attributes together have been proposed in the literature[51, 97]. In Guyon and Elisseeff [51], they divided the attribute selection methods into three types: wrappers, filters, and embedded methods. "Wrappers utilize the learning machine of interest as a black box to score subsets of variable according to their predictive power. Filters select subsets of variables as a pre-processing step, independently of the chosen predictor. Embedded methods perform variable selection in the process of training and are usually specific to given learning machines." However, it is always an issue choosing between selecting subsets of attributes that together have good predictive power, as opposed to ranking attributes according to their individual predictive power. In addition, several issues are found in wrapper and embedded methods such as computation complexity and overfitting. However, using the predictive machine as black box is appealing in terms of simplicity of use. As part of testing the proposed algorithms on gene expression datasets, we use another popular attribute selection algorithm called Relief [51, 97].

6.2 Gene Expression Datasets

This section gives a brief description of gene expression datasets used in our empirical evaluation.

1. Breast Cancer

   This dataset is made of patients outcome prediction for breast cancer. The original file is made of a training and testing datasets. The training data contains 78 patient samples, 34 of which are from patients who had developed distance metastases within 5 years (labelled as "relapse"), the rest 44 samples are from patients who remained healthy from the disease after their initial diagnosis for interval of at least 5 years (labelled as "non-relapse"). Correspondingly, there are 12 relapse and 7 non-relapse samples in the testing data set. The number of genes is 24481.

2. Central Nervous System

   Patients outcome prediction for central nervous system embryonal tumor. Survivors are patients who are alive after treatment whiles the failures are those who succumbed to their disease. The data set contains 60 patient samples, 21 are survivors (labelled as "Class1") and 39 are failures (labelled as "Class0"). There are 7129 genes in the dataset.

3. Colon Tumor

   Contains 62 samples collected from colon-cancer patients. Among them, 40 tumor biopsies are from tumors (labelled as "negative") and 22 normal (labelled as "positive") biopsies are from healthy parts of the colons of the same patients. Two thousand out of around 6500 genes were selected based on the confidence in the measured expression levels.

4. Lung Cancer (Dana-Farber Cancer Institute, Harvard Medical School)

   A total of 203 snap-frozen lung tumors and normal lung were analysized. The 203 specimens include 139 samples of lung adenocarcinomas (labelled as ADEN), 21 samples of squamous cell lung carcinomas (labelled as SQUA), 20 samples of pulmonary carcinoids (labelled as COID), 6 samples of small-cell lung carcinomas (labelled as SCLC) and 17 normal lung samples (labelled as NORMAL). Each sample is described by 12600 genes.

5. Ovarian Cancer (NCI PBSII Data)

   The goal of this experiment is to identify proteomic patterns in serum that distinguish ovarian cancer from non-cancer. This study is significant to women who have a high risk of ovarian cancer due to family or personal history of cancer. The proteomic spectra were generated by mass spectroscopy and the data set provided here is 6-19-02, which includes 91 controls (Normal) and 162 ovarian cancers. The raw spectral data of each sample contains the relative amplitude of the intensity at each molecular mass / charge (M/Z) identity. There are total 15154 M/Z identities. The intensity values were normalized according to the formula: NV = (V-Min)/(Max-Min), where NV is the normalized value, V the raw value, Min the minimum intensity and Max the maximum intensity. The normalization is done over all the 253 samples for all 15154 M/Z identities. After the normalization, each intensity value is to fall within the range of 0 to 1.

6. Prostate Cancer

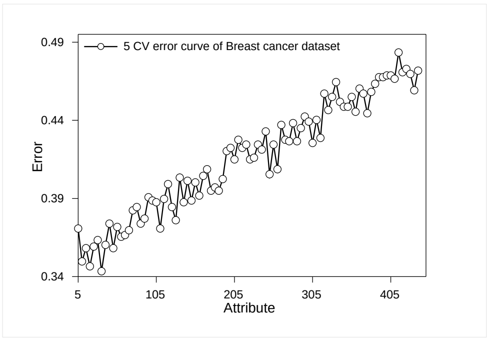
   (A) Tumor versus Normal classification: training set (from (1)) contains 52 prostate tumor samples and 50 non-tumor (labelled as "Normal") prostate samples with around 12600 genes. An independent set of testing samples from (2) is also prepared, which is from a different experiment and has a nearly 10-fold difference in overall microarray intensity from the training data. Besides, we have removed extra genes contained in the testing samples. In the above publication, the testing set is indicated to have 27 tumor and 8 normal samples. However, from our extraction, there are 25 tumor and 9 normal samples. (B) Prediction of clinical outcome: in this data set, 21 patients were evaluable with respect to recurrence following surgery with 8 patients having relapsed and 13 patients having remained relapse free (”non-relapse”) for at least 4 years.

Some datasets are given in two separate files as training and testing sets. We simply concatenated the training and testing files then use random train/test splits in the experiments.

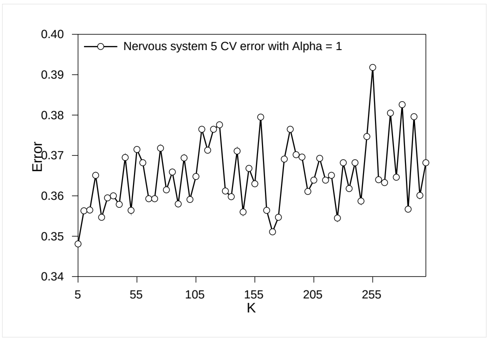
6.3 Experimental Setup

In Section 6.4.1, we use six benchmark gene expression datasets in order to evaluate the usefulness on the proposed algorithm in real world application. The three attribute filtering methods are implemented in WEKA. We evaluate the five classifiers on the first 5, 10, 20 30, 40 and 50 best ranked attributes. For these experiments we divide the datasets into a training set and a testing set. We use a stratified 10 fold Cross Validation (10CV) on the training set only to select the best values for α based on the average accuracy results of 15 experiments (model selection) for αRSC. The average classification accuracy of 30 experiments is calculated on each test set. For the comparison purpose, we continue with a single Decision tree (DT), NaiveBayes (NB), K nearest Neighbour (K-NN), Naive Bayes tree (NBtree) and non nested hyper-rectangle generalisation algorithm (NNge). For the ensembles, In Section 6.4.2, We use a stratified 10 fold Cross Validation (10CV) on the training set only to select the best values for α and κ based on the average accuracy results of 5 experiments (model selection). For comparison, we use Adabbost, Bagging, Random Comittee, Multiboost, Random Subspaces, Random Forest and Rotation Forest. All the ensembles use 100 classifiers. For the decision tree in the ensembles we keep the same default parameter as found in the WEKA package. The same applies for both Adaboost and Bagging. Random Subspaces uses half of the attributes of each dataset as suggested by its authors. In Random Forest we apply $\sqrt{(}k)$ rounded for the number of attributes which is 10.

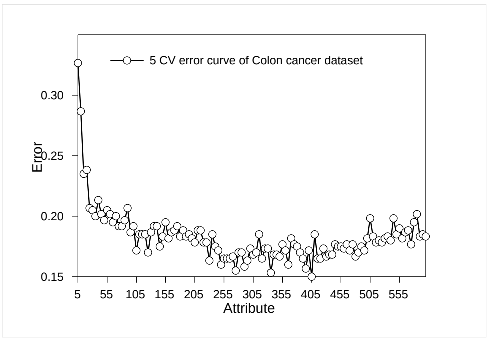
In the next two sections we assesses the usefulness of the αRSC and $\alpha RSSE$ on a real gene expression dataset using the three attribute selection methods.



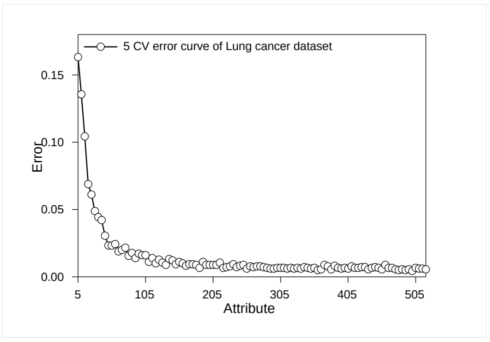
(a) $\alpha RSSE$ Learning Curve of of the Breast Cancer dataset



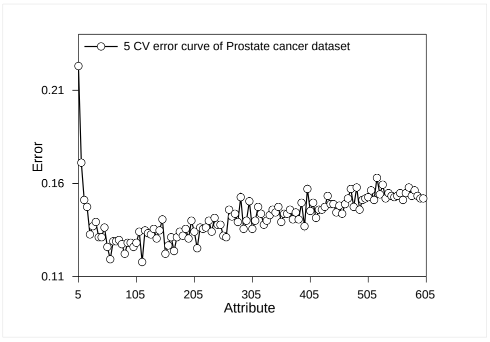
(b) $\alpha RSSE$ Learning Curve of the Nervous System dataset



(c) $\alpha RSSE$ Learning Curve of the Colon Cancer dataset



(d) $\alpha RSSE$ Learning Curve of the Lung Cancer Cancer dataset



(e) $\alpha RSSE$ Learning Curve of the Prostate Cancer dataset

Figure 6.1: αSCC Learning Curves on various Gene Expression datasets

6.4 Evaluation of Six Gene Expression Datasets with Three Attribute Filtering Methods

6.4.1 Performance of αRSC using χ^2, relief and Information Gain Ranked Attributes

The experiments produced 648 accuracy results over the 6 gene expression datasets using the 6 classifiers each on three attribute filtering methods (χ^2, Information Gain and Relief). In order to collate the results into a single table we calculated the Friedman ranking. Tables 6.1 shows the best performing classifier for each attribute filtering method. It is interesting to note that αRSC has ranked first on each attribute filtering method and, in most cases, has not ranked below third place. NNge and the Decision tree classifiers performed very badly in comparison to other classifier. These results suggest that αRSC performed very well over the 6 gene expression datasets on all the three attribute filtering methods.

The best results for each dataset regardless of cut-off points are shown in Tables 6.2, 6.3 and 6.4. In these tables we want to show which is the best performing classifier for each dataset since each classifier may perform badly on some cut-off while better on others. In addition, the main target of any classifiers is to find the best accuracy over a set of cut-offs. We also show Friedman average ranks for each attribute filtering method. The tables show that αRSC ranked 1^{st} for the χ^2, 2^{nd} for Relief and 3^{rd} for the Information gain filtering methods.

The overall ranking results of the three attribute filtering methods is calculated by summing the average ranks of the three tables as shown in Table 6.5. αRSC has ranked first while K-NN ranked 2^{nd}. These results show that αRSC is a good classifier for these gene expression datasets, and that it works well with attribute filters.

6.4.2 Performance of $\alpha RSSE$ using χ^2, Relief and Information Gain Ranked Attributes

We trained the $\alpha RSSE$ ensemble on the gene expression datasets using the full attribute sets. Figure 6.1 shows the 5CV learning curves which also shows the degree of difficulty of each dataset.

Tables 6.6, 6.7 and 6.8 show the best performing classifier for each attribute filtering method. It is interesting to note that $\alpha RSSE$ has ranked first on two attribute filtering methods (χ^2 and IF) and ranked third place for Relief filtering method. Random Subspaces,

Table 6.1: The ranking based on classification accuracy of six datasets of αRSC, K-Nearest neighbour (K-NN), Decision tree (J48), Naive Bayes tree (NBT), NaiveBayes (NB) and Non-nested Generalised Hyper-rectangle (NNge) using average results of 30 different runs on χ^2, Information Gain (IG) and Relief.

Algorithms	αRSC	DT	K-NN	NB	Nbtree	NNge
ranked all dataset χ^2						
top5	3	5	6	4	1	2
top10	1	6	2	5	3	4
top20	3	6	4	1	2	5
top30	3	6	2	1	5	4
top40	3	6	1	2	4	5
top50	2	6	1	4	3	5
Avg	2.5	5.83	2.67	2.83	3	4.17
Total ranks	1	6	2	3	4	5
ranked all dataset IG						
top5	2	6	5	4	1	3
top10	2	6	1	4	3	5
top20	3	6	1	4	2	5
top30	5	6	1	2	3	4
top40	4	6	1	5	3	2
top50	1	6	2	5	3	4
Avg	2.83	6	1.83	4	2.5	3.83
Total ranks	3.5	6	1	5	2	3.5
ranked all dataset Relief						
top5	2	6	4	5	5	3
top10	1	6	3	2	5	4
top20	1	6	3	2	5	4
top30	1	6	3	2	4	5
top40	2	6	3	1	5	4
top50	3	6	1.50	1.50	4	5
Avg	1.67	6	2.92	2.25	4.67	4.17
Total ranks	1	6	3	2	5	4

Table 6.2: The best test set accuracy (in %) of αRSC, K-Nearest neighbour (K-NN), Decision tree (J48), Naive Bayes tree (NBT), NaiveBayes (NB) and Non-nested Generalised Hyper-rectangle (NNge) using average results of 30 different runs on χ^2. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Dataset	αRSC	NBTree	K-NN	NB	NNge	DT
Breast Cancer	77.58	76.16	75.35	71.11	71.01	72.42
Prostate	91.01	90.87	94.35	70.00	89.35	90.22
Lung Cancer	99.13	99.23	99.07	100.00	99.95	95.63
Ovarian	98.86	97.96	99.33	98.59	98.55	97.10
Colon Tumor	85.24	88.10	84.29	87.46	84.29	83.81
Central Nervous	80.33	80.67	78.83	78.17	74.00	76.67
Average Rank	2.33	2.50	2.92	3.50	4.58	5.17
ranks	1	2	3	4	5	6

Table 6.3: The best test set accuracy (in %) of αRSC, K-Nearest neighbour (K-NN), Decision tree (J48), Naive Bayes tree (NBT), NaiveBayes (NB) and Non-nested Generalised Hyper-rectangle (NNge) using average results of 30 different runs on Relief (RL). Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Dataset	NB	αRSC	K-NN	NNge	NBTree	DT
Breast Cancer	81.62	77.37	80.40	73.84	74.65	71.52
Prostate	76.09	91.96	95.07	87.75	89.13	89.71
Lung Cancer	99.29	99.23	98.31	99.07	98.69	95.96
Ovarian	98.78	97.88	99.18	98.59	97.84	97.10
Colon Tumor	85.08	86.03	80.79	82.7	82.86	79.68
Central Nervous	78.33	77.17	76.83	70.83	70.67	71.17
Average Rank	2.17	2.33	2.83	4.17	4.33	5.17
Ranks	1	2	3	4	5	6

Table 6.4: The best test set accuracy (in %) of αRSC, K-Nearest neighbour (K-NN), Decision tree (J48), Naive Bayes tree (NBT), NaiveBayes (NB) and Non-nested Generalised Hyper-rectangle (NNge) using average results of 30 different runs on Information Gain (IG). Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Dataset	K-NN	Nbtree	αRSC	NNge	NB	DT
Breast Cancer	75.35	76.87	78.38	69.90	69.9	72.63
Prostate	90.51	88.99	89.49	87.61	67.25	89.71
Lung Cancer	99.18	99.67	99.34	100.00	100.00	95.63
Ovarian	99.53	98.04	98.90	98.59	98.59	97.06
Colon Tumor	85.40	86.51	85.87	84.92	84.44	82.22
Central Nervous	77.83	82.83	74.00	75.33	75.67	74.50
Average Ranks	2.50	2.67	3.00	3.75	3.92	5.17
Ranks	1	2	3	4	5	6

Table 6.5: All Ranks over the three attribute filtering methods

Classifiers	Sum Ranks	All Ranks
αRSC	7.00	1
K-NN	7.50	2
Nbtree	8.75	3
NB	11.42	4
NNge	12.83	5
DT	15.50	6

Adaboost and BAgging performed badly in comparison to other classifier. These results suggest that $\alpha RSSE$ performed very well over the 6 gene expression datasets on all the three attribute filtering methods.

The best results for each dataset regardless of the filtering methods are shown in Tables 6.9. In this tables we want to show which is the best performing classifier for each dataset since each classifier may perform badly on some filtering method but better on others. We also show Friedman average ranks for each attribute filtering method. The tables show that $\alpha RSSE$ ranked 1^{st}, Random Forest 2^{nd} and Rotation Forest 3^{rd}. Both Random Forest and $\alpha RSSE$ have similar average ranks. These experiments show that $\alpha RSSE$ performed very well on these gene expression datasets and produced similar results to those of Rotation Forest and Random Forest.

Table 6.6: The best test set accuracy (in %) of $\alpha RSSE$, Rotation Forest (RotF), Random Subspace (RandS), Random Forest (RandF), Adaboost, bagging and MultiBoostAB (Multi) using average results of 30 different runs on χ^2. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Dataset	$\alpha RSSE$	RotF	RandS	RandF	Adaboost	Bagging	Multi
Breast Cancer	82.93	79.60	76.26	80.91	79.19	78.99	78.79
Central Nervous	77.83	76.83	74.33	80.33	76.33	76.17	76.50
Colon Tumor	85.87	86.19	83.49	84.13	82.38	83.65	82.86
Lung Cancer	99.34	99.34	95.03	99.34	97.81	97.21	97.87
Ovarian	99.18	99.80	97.88	98.98	97.73	97.84	97.73
Prostate	94.13	93.70	91.30	94.57	91.23	91.38	91.09
Average Ranks	6.17	5.83	2.17	6.00	2.42	3.00	2.42
Ranks	1	3	7	2	5.5	4	5.5

Table 6.7: The best test set accuracy (in %) of $\alpha RSSE$, Rotation Forest (RotF), Random Subspace (RandS), Random Forest (RandF), Adaboost, bagging and MultiBoostAB (Multi) using average results of 30 different runs on Information Gain.

Dataset	$\alpha RSSE$	RotF	RandS	RandF	Adaboost	Bagging	Multi
Breast Cancer	85.15	79.39	77.47	83.94	79.49	80.10	79.80
Central Nervous	79.17	76.50	73.50	80.00	75.67	76.17	76.00
Colon Tumor	86.98	84.76	82.54	84.44	82.70	82.54	82.38
Lung Cancer	99.34	99.34	94.75	99.34	97.76	97.16	97.81
Ovarian	99.25	99.76	98.00	98.86	97.73	97.88	97.73
Prostate	93.77	93.48	91.74	93.62	91.09	92.32	90.80
Average Ranks	6.50	5.17	2.08	5.83	2.58	3.42	2.42
Ranks	1	3	7	2	5	4	6

Table 6.8: The best test set accuracy (in %) of $\alpha RSSE$, Rotation Forest (RotF), Random Subspace (RandS), Random Forest (RandF), Adaboost, bagging and MultiBoostAB (Multi) using average results of 30 different runs on Relief. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Dataset	$\alpha RSSE$	RotF	RandS	RandF	Adaboost	Bagging	Multi
Breast Cancer	80.20	79.19	72.42	78.18	73.74	74.85	73.23
Central Nervous	76.00	75.50	72.17	76.00	74.00	72.00	73.33
Colon Tumor	83.65	84.76	80.63	83.33	79.37	83.17	79.68
Lung Cancer	99.34	99.23	94.75	98.91	97.43	96.61	97.49
Ovarian	98.43	99.37	98.04	98.90	97.61	97.69	97.61
Prostate	89.13	93.33	91.67	93.62	93.41	89.71	93.26
Average Ranks	5.42	6.00	2.33	5.75	3.08	2.67	2.75
Ranks	3	1	7	2	4	6	5

Table 6.9: The best test set accuracy (in %) of $\alpha RSSE$, Rotation Forest (RotF), Random Subspace (RandS), Random Forest (RandF), Adaboost, bagging and MultiBoostAB (Multi) of the three attribute ranking methods. Average Ranks stands for Friedman average ranks and Ranks stands for Friendam ranks.

Dataset	$\alpha RSSE$	RotF	RandS	RandF	Adaboost	Bagging	Multi
Breast Cancer	84.04	79.60	77.47	83.94	79.49	80.10	79.80
Central Nervous	79.17	76.83	74.33	80.33	76.33	76.17	76.5
Colon Tumor	86.98	86.19	83.49	84.44	82.70	83.65	82.86
Lung Cancer	99.34	99.34	95.03	99.34	97.81	97.21	97.87
Ovarian	99.18	99.76	98.00	98.98	97.73	97.88	97.73
Prostate	94.13	93.70	91.74	94.57	93.41	92.32	93.26
Average Ranks	6.33	5.33	1.83	6.00	2.42	3.00	3.08
Ranks	1	3	7	2	6	5	4

6.5 Chapter Summary and Future Research

In this chapter, we used six gene expression datasets to evaluate the proposed classifiers. We showed that αRSC performs as good as various popular classifiers using three ranking methods. We also showed that $\alpha RSSE$ performs as good as Rotation Forest and Random Forest, and better than Random Subspace.

Chapter 7

Conclusion

7.1 Main Investigations and Findings of this Thesis

This thesis evaluated the sphere cover in supervised learning. In the first part of our investigation, we randomised covers produced by a sphere cover and introduced a simple classifier which we called αRSC. We studied the relationships between classification accuracy and pruning parameter. We used various benchmark datasets in order to assess the classification accuracy of αRSC classifier. We found that searching for the best pruning value produced classification accuracy that are similar to those of K-NN. In order to understand the generalization error of the Randomized Sphere Cover classifier, we used bias/variance decomposition.

In particular, we studied the αRSC classifier using the compression scheme. The results produced from this study shows an intrinsic relationships between training accuracy, compression set, and some other specific information about the classifier. These three factors are combined in a PAC compression bound to estimate the lowest generalization error in relation to the pruning parameter α. The compression scheme study prompted us to investigate other methods that produce the smallest compression set. We used a Gaussian kernel function with the αRSC classifier. We used two datasets to evaluate the proposed method. The results showed a smaller compression set may be produced using a kernel method.

Next, we introduced an ensemble method for the αRSC classifier. Two issues are discussed for this inquiry. First, what is the best method that randomize (perturb) covers while keeping accurate classifiers (αRSE)? Second, what is the best way to further randomise classifiers using an ensemble method ($\alpha\beta RSE$)? We used various benchmark datasets in order to evaluate the classification accuracy of the proposed ensembles. We compared the results to those of known ensemble methods. We continued with the bias/variance analysis in order to compare the results with those of a single Randomized Sphere Cover classifier. The bias/variance results showed that the proposed ensemble method reduces mainly the variance.

Finally, we introduced a Subspace Randomized Sphere Cover ensemble ($\alpha RSSE$). The ensemble builds covers using random subsets of attributes. We used twenty five datasets in order to evaluate the proposed subspace method. We showed that the classification accuracies of our subspace method are competitive with those of Random Forest and Rotation Forest. We also found the classification accuracies of the proposed subspace method are superior to those of Bagging, Adaboost and Random Subspace. We showed the bias/variance decomposition of the error and made a comparative analysis with a single Randomized Sphere Cover classifier and our ensemble method. We showed that bias and unbiased variance was the main reason for the overall reduction in classification error of $\alpha RSSE$.

In order to verify the good performance of our proposed classifiers, we used six real gene expression datasets. We used three attribute ranking methods on these gene expression dataset in order to assess the classifiers proposed in this thesis. We showed that the αRSC classifier produced results similar to those of K-NN and Naive Bayes Trees. We showed that our classier shows consistent results over the three ranking methods while producing high classification accuracy in comparison with other classifiers. We also showed that the $\alpha RSSE$ classifier produced results similar to those of Rotation Forest and Random Forest. We showed that our ensemble method shows consistent results over the three ranking methods while producing high classification accuracy in comparison with other ensemble methods.

7.2 Limitations of this Thesis

In Chapter 3, we produced no comparative study that includes the bias/variance decomposition. It would have been interesting to employ a direct comparison using bias/variance decomposition with other classification algorithms, for instance, nearest neighbour classifier and decision trees. In Chapter 4, we discussed the limitations of the existing sample compression bounds. However, we investigated a simple way to represent the message string (the added information of a classifier). Searching a better way that represent the added information might give much tighter bound. In Chapter 5, the limitations consists of lack of comparisons with ensemble methods that use base classifiers other than decision trees. Indeed, a comparison with ensemble methods that employ Nearest Neighbour, Neural Networks and SVM as base classifiers would have given us a better evaluation. In addition, other sampling method could have been used based on the margins of examples and some diversity measures. Indeed, diversity measures may have helped in selecting the most diverse classifiers for the ensemble. In Chapter 6, we only used the classification accuracy on gene expression datasets for the comparisons. Receiver Operating Characteristic (ROC) gives better performance evaluation for medical datasets.

7.3 Summary of Contributions

- Chapter 3: We proposed a Randomized Sphere Cover Classifier using a single regularization parameter. We investigated its classification performance and made a comparison with several known classifiers. We used the bias/variance decomposition in order to analyze the generalization error of the proposed classifier.
- Chapter 4: We investigated the Randomized Sphere Cover Classifier using the compression scheme, and proposed a new method for generating covers using the kernel trick.
- Chapter 5: We proposed an ensemble of Randomized Sphere Cover Classifier taking into consideration the geometrical property of the base classifier. We used thirty dataset to evaluate the ensemble classification accuracy and compared the results to those of known ensemble methods. We compared the bias/variance decomposition results to those of a single Randomized Sphere Cover Classifier. We proposed an ensemble build in the subspaces using the Randomized Sphere Cover Classifier. We investigated the classification accuracy of the proposed subspace ensemble on real gene expression datasets. We studied the bias/variance decomposition in order to make a comparison with the previous proposed classifiers.
- Chapter 6: Finally, we used three attribute ranking methods on six gene expression datasets in order to evaluate and compare the classifiers proposed in this thesis with other well known classifiers and ensemble methods.

7.4 Future Considerations

For the ensemble methods produced we used the majority vote. Other combination methods showed to work better than majority vote and may improve our results. We used several diversity measures during the course of this research with the ensemble proposed in Chapter 5. In future research diversity measures will be employed to select base classifier according to their dissimilarity with other members of the ensemble. We would have liked to compare $\alpha RSSE$ results with those of FASBIR and SFS which we will be left for future research.

References

[1] David W. Aha, Dennis Kibler, and Marc K. Albert. Instance-based learning algorithms. *Machine Learning*, **6**:37–66, 1991. 7, 8, 74

[2] M.C. Ainslie and J.S. Sánchez. Space partitioning for instance reduction in lazy learning algorithms. In *Second Workshop on Integration and Collaboration Aspects of Data Mining, Decision Support and Meta-Learning*, pages 13–18, 2002. 8

[3] Ethem Alpaydin. Voting over multiple condensed nearest neighbors. *Artif. Intell. Rev.*, **11**(1-5):115–132, 1997. 34

[4] Amir F. Atiya, Sherif Hashem, and Hatem A. Fayed. Pattern classification using a set of compact hyperspheres. In Irwin King, Jun Wang, Laiwan Chan, and DeLiang L. Wang, editors, *Neural Information Processing, 13th International Conference, Hong Kong, China, Proceedings, Part II*, **4233**, pages 116–123. Springer, 2006. 18, 53

[5] Arindam Banerjee. On bayesian bounds. In *Proceedings of the twenty third international conference on Machine learning (ICML)*, pages 81–88. ACM, 2006. 63

[6] Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, **36**(1-2):105–139, 1999. 20, 27

[7] Stephen D. Bay. Combining nearest neighbor classifiers through multiple feature subsets. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 37–45, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. 26

[8] Sergio Bermejo and Joan Cabestany. Local averaging of ensembles of lvq-based nearest neighbor classifiers. *Appl. Intell.*, **20**(1):47–58, 2004. 34

[9] Vladimir N. Vapnik Bernhard E. Boser, Isabelle M. Guyon. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 144–152, New York, NY, USA, 1992. ACM. 55

[10] James C. Bezdek and Ludmila I. Kuncheva. Some notes on twenty one (21) nearest prototype classifiers. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 1–16, London, UK, 2000. Springer-Verlag. 8, 34

[11] James C. Bezdek and Ludmila I. Kuncheva. Nearest prototype classifier designs: An experimental study. *Int. J. Intell. Syst*, **16**(12):1445–1473, 2001. 8, 34

[12] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, **2**, 2002. 34

[13] Leo Breiman. Bagging predictors. *Machine Learning*, **24**(2):123–140, 1996. 21, 22

[14] Leo Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Departament, University of California, Berkeley, 1996. 27, 28, 74

[15] Leo Breiman. Half&half bagging and hard boundary points. Technical Report 30, Statistics Departament, University of California, Berkeley, sep 1998. 22, 75

[16] Leo Breiman. Random forests. *Machine Learning*, **45(1)**:5–32, 2001. 26, 74

[17] Henry Brighton and Chris Mellish. Advances in instance selection for instance-based learning algorithms. *Data Min. Knowl. Discov*, **6**(2):153–172, 2002. 8, 13, 34

[18] Carla E. Brodley and Mark A. Friedl. Identifying and eliminating mislabeled training instances. In *Proceedings of the thirteenth national conference on Artificial intelligence - Volume 1*, AAAI'96, pages 799–805. AAAI Press, 1996. 13

[19] Gavin Brown. *Diversity in Neural Network Ensembles*. PhD thesis, School of Computer Science, University of Birmingham, 2004. 20

[20] Gavin Brown and Jeremy Wyatt. Negative correlation learning and the ambiguity family of ensemble methods. In *Proceedings of the 4th international conference on Multiple classifier systems*, MCS'03, pages 266–275, Berlin, Heidelberg, 2003. Springer-Verlag. 25

[21] Adam Cannon, J. Mark Ettinger, Don Hush, and Clint Scovel. Machine learning with data dependent hypothesis classes. *J. Mach. Learn. Res.*, **2**:335–358, March 2002. 10, 11, 13, 55, 57

[22] Adam H. Cannon and Lenore J. Cowen. Approximation algorithms for the class cover problem. *Annals of Mathematics and Artificial Intelligence*, **40**:215–223, March 2004. 13, 34

[23] Lenore J. Cowen and Carey E. Priebe. Randomized non-linear projections uncover high-dimensional structure. Technical Report 11, Department of Mathematical Sciences; Johns Hopkins University; Baltimore , MD 21218, dec 1997. 9

[24] Koby Crammer, Ran Gilad-Bachrach, Amir Navot, and Naftali Tishby. Margin analysis of the lvq algorithm. In *The Neural Information Processing Systems (NIPS)*, pages 462–469, 2002. 24

[25] Felipe Cucker and Steve Smale. On the mathematical foundations of learning. *Bulletin of the American Mathematical Society*, **39**:1–49, 2002. 55

[26] B. V. Dasarathy, editor. *Nearest Neighbor Pattern Classification Techniques*. IEEE Computer Society Press, 1991. 75

[27] Sanjoy Dasgupta and Philip M. Long. Boosting with diverse base classifiers. In *in Learning Theory and Kernel Machines, 16th Annual Conference on Learning Theory and 7th Kernel Workshop, Lecture Notes in Artificial Intelligence 2777*, pages 273–287. Springer, 2003. 20

[28] Carey E. Priebe David J. Marchette, Edward J. Wegman. A fast algorithm for approximating the dominating set of a class cover catch digraph. In *technical report, JHU DMS TR 635*. galaxy.gmu.edu, 2003. 15, 34, 35

[29] Carey E. Priebe David J. Marchette Jason G. DeVinney and Diego A. Socolinsky. Classification using class cover catch digraphs. *Journal of Classification*, **20**(1):3–23, 2003. 11, 13, 15, 37

[30] Jason DeVinney and Carey E. Priebe. A new family of proximity graphs: Class cover catch digraphs. *Discrete Applied Mathematics*, **154**(14):1975–1982, 2006. 11

[31] Jason G. DeVinney. *The Class Cover Problem and Its Application in Pattern Recognition*. PhD thesis, The Johns Hopkins University, USA, 2003. 11

[32] Thomas G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, **40**(2):139–157, 2000. 20

[33] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *J. Artif. Int. Res.*, **2**:263–286, January 1995. 27

[34] Pedro Domingos. A unified bias-variance decomposition for zero-one and squared loss. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 564–569, 2000. 6, 27, 29, 30

[35] André Elisseeff, Theodoros Evgeniou, and Massimiliano Pontil. Stability of randomized learning algorithms. *Journal of Machine Learning Research*, **6**:55–79, 2005. 20, 28

[36] Theodoros Evgeniou and Massimiliano Pontil. Statistical learning theory: a primer. *International Journal of Computer Vision*, **38**:9–13, 2000. 55

[37] Theodoros Evgeniou, Massimiliano Pontil, and André Elisseeff. Leave one out error, stability, and generalization of voting combinations of classifiers. *Machine Learning*, **55**(1), 2004. 34

[38] Sally Floyd and Manfred K. Warmuth. Sample compression, learnability, and the vapnik-chervonenkis dimension. *Machine Learning*, **21**(3):269–304, 1995. 4, 55, 56, 57

[39] A. Frank and A. Asuncion. UCI Machine Learning Repository, University of California, Irvine, School of Information and Computer Sciences, http://archive.ics.uci.edu/ml, 2010. 5

[40] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 148–156, 1996. 20

[41] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *COLT '96: Proceedings of the ninth annual conference on Computational learning theory*, pages 325–332. ACM, 1996. 20

[42] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, **55**, 1997. 22

[43] Jerome H. Friedman. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *J. Data Mining and Knowledge Discovery*, **1**(1):55–77, 1997. 27

[44] Jerome H. Friedman and Usama Fayyad. On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, **1**:55–77, 1997. 27

[45] Byron J. Gao and Martin Ester. Right of inference: Nearest rectangle learning revisited. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *ECML*, **4212** of *Lecture Notes in Computer Science*, pages 638–645. Springer, 2006. 9

[46] Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural Comput.*, **4**:1–58, January 1992. 27, 74

[47] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine Learning*, **63**(1):3–42, 2006. 20, 29, 73, 74

[48] Yves Grandvalet, Stéphane Canu, and Stéphane Boucheron. Noise injection: Theoretical prospects. *Neural Computation*, **9**(5):1093–1108, 1997. 20, 25

[49] Lei Gu and Hui-Zhong Wu. Applying a novel decision rule to the sphere-structured support vector machines algorithm. *Neural Computing and Applications*, **18**(7):675, 2009. 53

[50] Lei Gu, Hui zhong Wu, and Liang Xiao. A novel classification algorithm based on fuzzy kernel multiple hyperspheres. In *FSKD '07: Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007) Vol.2*, pages 114–118. IEEE Computer Society, 2007. 53

[51] Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, **3**:1157–1182, March 2003. 26, 100, 101

[52] Jiawei Han and Micheline Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000. 74

[53] Jakob V. Hansen and Tom Heskes. General bias/variance decomposition with target independent variance of error functions derived from the exponential family of distributions. In *in 15th International Conference on Pattern Recognition*, pages 207–210, 1999. 27

[54] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, **12**:993–1001, October 1990. 20

[55] David Haussler. Quantifying inductive bias: AI learning algorithms and valiant's learning framework. *Artificial Intelligence*, **36**(2):177–221, 1988. 57

[56] Ralf Herbrich. *Learning Kernel Classifiers, Theory and Algorithms*. MIT Press Cambridge, MA, USA, 2002. 2, 55, 56, 57, 66, 67

[57] Haym Hirsh and William W. Cohen. Learning from data with bounded inconsistency: theoretical and experimental results. In *Proceedings of a workshop on Computational learning theory and natural learning systems (vol. 1) : constraints and prospects: constraints and prospects*, pages 355–380, Cambridge, MA, USA, 1994. MIT Press. 75

[58] T. K. Ho. Nearest neighbors in random subspaces. In *Proceedings of the 2nd Int'l Workshop on Statistical Techniques in Pattern Recognition*, pages 640–648, Sydney, Australia, 1998. 26, 74

[59] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions Pattern Analysis and Machine Intelligence*, **20**(8):832–844, 1998. 26, 74

[60] T. K. Ho. Data complexity analysis for classifier combination. In *Multiple Classifier Systems*, pages 53–67, 2001. 34

[61] T. K. Ho. Geometrical complexity of classification problems, 2004. Comment: Proceedings of the 7th Course on Ensemble Methods for Learning Machines at the International School on Neural Nets, E.R. Caianiello. 44, 53, 75

[62] T. K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, **24**:289–300, March 2002. 34

[63] Michael J. Hudak. RCE networks: An experimental investigation. In *IEEE International Joint Conference on Neural Networks*, **I**, pages 849–854. IEEE, 1991. 18, 53

[64] Zakria Hussain, Francois Laviolette, Mario Marchand, John Shawe-Taylor, Spencer Charles Brubaker, and Matthew D. Mullin. Revised loss bounds for the set covering machine and sample-compression loss bounds for imbalanced data. *Journal of Machine Learning Research*, **8**:2533–2549, 2007. 63

[65] Gareth James. Variance and bias for general loss functions. *Machine Learning*, **51**:115–135, 2003. 27

[66] Gareth James and Trevor Hastie. Generalizations of the bias/variance decomposition for prediction error. Technical report, Department of Statistics, Stanford University, Stanford, CA., 1997. 27

[67] Demšar Janez. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, **7**:1–30, 2006. 39, 80

[68] Christian Jutten et al. Publications from ELENA partners. Technical report, Elena-NervesII, Enhanced Learning for Evolutive Neural Architecture, ESPRIT-Basic Research Project Number 6891, 1995. 18

[69] Matti Kääriäinen and John Langford. A comparison of tight generalization error bounds. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML), Bonn, Germany, August 7-11, 2005*, **119**, pages 409–416. ACM, 2005. 6

[70] Sang-Woon Kim and B. John Oommen. A brief taxonomy and ranking of creative prototype reduction schemes. *Pattern Anal. Appl*, **6**(3), 2003. 8, 34

[71] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**:226–239, March 1998. 20

[72] Eun Bae Kong and Thomas G. Dietterich. Error-correcting output coding corrects bias and variance. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, pages 313–321. Morgan Kaufmann, 1995. 27

[73] L. I. Kuncheva, C. Whitaker, C. Shipp, and R. Duin. Limits on the majority vote accuracy in classifier fusion. *Pattern Analysis and Applications*, **6**(1):22–31, 2003. 21

[74] L. I. Kuncheva and C. J. Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, **51**:181–207, 2003. 20, 27, 73

[75] Ludmila I. Kuncheva. A theoretical study on six classifier fusion strategies. *PAMI*, **24**(2):281–286, 2002. 21

[76] Ludmila I. Kuncheva. That elusive diversity in classifier ensembles. In *Lecture Notes in Computer Science*, pages 1126–1138, 2003. 27

[77] Ludmila. I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley Press, 2004. 21, 74

[78] Ludmila I. Kuncheva. Diversity in multiple classifier systems. *Information Fusion*, **6**(1):3–4, 2005. 27, 73

[79] Ludmila I. Kuncheva and James C. Bezdek. An integrated framework for generalized nearest prototype classifier design. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **6**(5):437–458, 1998. 7, 8

[80] Stephen Sengwah Kwek. Geometric concept learning and related topics. Technical Report 1980, Department of Computer Science, University of Illinois at Urbana-Champaign, 1996. 75

[81] John Langford. Tutorial on practical prediction theory for classification. *Journal of Machine Learning Research*, **6**:273–306, 2005. 63

[82] Nick Littlestone and Manfred K. Warmuth. Relating data compression and learnability. unpublished manuscript, University of California Santa Cruz, 1986. 4, 55, 56

[83] Shuang Liu, Yongkui Liu, Bo Wang, and Xiwei Feng. An improved hyper-sphere support vector machine. *International Conference on Natural Computation*, **1**:497–500, 2007. 53

[84] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, **12**(10):1399–1404, 1999. 25

[85] Li T Zhang C Ogihara M. A comparative study of feature selection and multiclass classification methods for tissue classification based on gene expression. *Bioinformatics*, **20**:2429–2437, 2004. 100

[86] Oded Maimon and Lior Rokach. *The Data Mining and Knowledge Discovery Handbook*. Springer, 2005. 7, 29, 74

[87] Man Wai Mak and Kin Wai Cho. Genetic evolution of radial basis function centers for pattern classification. In *In Proc. of The 1998 IEEE International Joint Conference on Neural Networks*, pages 669–673, 1998. 19

[88] Mario Marchand and John Shawe-Taylor. The set covering machine. *Journal of Machine Learning Research*, **3**:723–746, 2002. 56

[89] Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Research*, **6**:427–451, 2005. 56

[90] Mario Marchand and Marina Sokolova. Learning with decision lists of data-dependent features. *Journal of Machine Learning Research*, **6**:427–451, 2005. 57, 59, 63

[91] David J. Marchette. *Random Graphs for Statistical Pattern Recognition*. Wiley-Interscience, 2004. 13, 38

[92] Brent Martin. *Instance-Based learning : Nearest Neighbor With Generalization*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1995. 9, 39

[93] David A. McAllester. Some pac-bayesian theorems. *Machine Learning*, **37**:355–363, 1999. 63

[94] David A. McAllester. Simplified PAC-bayesian margin bounds. In *COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers*, 2003. 63

[95] Ron Meir and Gunnar Rätsch. An introduction to boosting and leveraging. In Shahar Mendelson and Alex J. Smola, editors, *Machine Learning Summer School*, **2600** of *Lecture Notes in Computer Science*, pages 118–183. Springer, 2002. 20, 25

[96] Prem Melville and Raymond J. Mooney. Creating diversity in ensembles using artificial data. *Information Fusion*, **6**(1):99–111, 2005. 20, 25

[97] Luis Carlos Molina, Lluís Belanche, and Àngela Nebot. Feature selection algorithms: A survey and experimental evaluation. In *ICDM*, pages 306–313. IEEE Computer Society, 2002. 26, 74, 99, 100, 101

[98] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Comput.*, **1**:281, 1989. 19

[99] Fabrice Muhlenbach, Lallich Stéphane, and Djamel A Zighed. Identifying and handling mislabelled instances. *J. Intell. Inf. Syst.*, **22**(1):89–109, 2004. 13

[100] M.T. Musavi, K. Faris, K.H. Chan, and W. Ahmed. A clustering algorithm for implementation of rbf technique. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, **ii**, page 972 vol.2, jul 1991. 19

[101] Anand M. Narasimhamurthy. Evaluation of diversity measures for binary classifier ensembles. In *Multiple Classifier Systems*, pages 267–277, 2005. 20

[102] David Opitz. Feature selection for ensembles. In *Proceedings of 16th National Conference on Artificial Intelligence (AAAI)*, pages 379–384, 1999. 26

[103] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, **11**:169–198, 1999. 20

[104] Nemenyi PB. *Distribution-free multiple comparisons*. PhD thesis, Princeton University, New Jersey, 1963. 39

[105] Michael P. Perrone and Leon N. Cooper. *Coulomb potential learning*. MIT Press, Cambridge, MA, USA, 1998. 17

[106] Petra Philips. *Data-dependent analysis of learning algorithms*. PhD thesis, Computer Science, 2005. 57

[107] C. Priebe, J. DeVinney, and D. Marchette. the distribution of the domination number for random class cover catch digraphs. *Statistics and Probability Letters*, **55**:239–246, 2001. 10, 11

[108] J. Quinlan. Induction of decision trees. *Machine Learning*, **1**:81–106, 1986. 9, 39, 80

[109] G. Rätsch, T. Onoda, and K.R. Mueller. Soft margins for adaboost. *Machine Learning*, **42**(3):287, 2001. 25

[110] Gunnar Rätsch and Manfred K. Warmuth. Efficient margin maximizing with boosting. *Journal of Machine Learning Research*, **6**:2131–2152, 2005. 25

[111] Y. Raviv and N. Intrator. Bootstrapping with noise: An effective regularization technique. *Connection Science, Special issue on Combining Estimators*, **8**:356–372, 1996. 25

[112] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, **28**(10):1619–1630, 2006. 74

[113] Cynthia Rudin, Ingrid Daubechies, and Robert E.Schapire. On the dynamics of boosting. In Sebastian Thrun, Lawrence K. Saul, and Bernhard Schölkopf, editors, *The Neural Information Processing Systems (NIPS)*, page 2004. MIT Press, 2003. 25

[114] Cynthia Rudin, Robert E. Schapire, and Ingrid Daubechies. Analysis of boosting algorithms using the smooth margin function. *Ann. Statist.*, **35**(6):2723–2768, 2007. 25

[115] Steven Salzberg. A nearest hyperrectangle learning method. *Machine Learning*, **6**:251–276, May 1991. 8

[116] Robert E. Schapire. Theoretical views of boosting and applications. In *Proceedings of the 10th International Conference on Algorithmic Learning Theory*, ALT '99, pages 13–25, London, UK, 1999. Springer-Verlag. 20

[117] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. 14th International Conference on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997. 22, 24

[118] Bernhard Scholkopf. The kernel trick for distances. In *The Neural Information Processing Systems (NIPS)*, pages 301–307, 2000. 67

[119] Marc Sebban, Richard Nock, and Stéphane Lallich. Boosting neighborhood-based classifiers. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML)*, pages 505–512, 2001. 33

[120] Marc Sebban, Richard Nock, and Stéphane Lallich. Stopping criterion for boosting-based data reduction techniques: from binary to multiclass problem. *Journal of Machine Learning Research*, **3**:863–885, 2002. 34

[121] M. Skurichina, S. Raudys, and R. P. W. Duin. K-nearest neighbors directed noise injection in multilayer perceptron training. *IEEE Transactions on Neural Networks*, **11**(2):504–511, 2000. 25

[122] M. W. Skurichina and R. P. W. Duin. Bagging, boosting and the random subspace method for linear classifiers. *Pattern Analysis & Applications*, **5**(2):121–135, 2002. 74

[123] Marina Skurichina, Ludmila Kuncheva, and Robert P.W. Duin. Bagging and boosting for the nearest mean classifier: Effects of sample size on diversity and accuracy. In *Multiple Classifier Systems*, pages 62–71, 2002. 20

[124] Alexander J. Smola, Bernhard Scholkopf Peter Bartlett, and Dale Schuurmans. *Advances in Large Margin Classifiers.* The MIT Press. Cambridge, 2000. 24, 55, 66

[125] E. Tang, P. Suganthan, and X. Yao. An analysis of diversity measures. *Machine Learning*, **65**:247–271, 2006. 20, 27, 73

[126] Robert Tibshirani. Bias, variance and prediction error for classification rules. Technical report, University of Toronto, Dept. of Statistics, Toronto, 1996. 27

[127] Robert Tibshirani. Bias, variance and prediction error for classification rules. Technical report, University of Toronto, Dept. of Statistics, Toronto, 1996. 27

[128] Alexey Tsymbal, Mykola Pechenizkiy, and Padraig Cunningham. Diversity in search strategies for ensemble feature selection. *Information Fusion*, **6**(1):83–98, 2005. 26, 27, 74

[129] Giorgio Valentini. *Ensemble methods based on bias-variance analysis*. PhD thesis, DISI - Dipartimento di Informatica e Scienze dell' Informazione, Universita, di Genova, 2003. 34

[130] Giorgio Valentini. An experimental bias-variance analysis of svm ensembles based on resampling techniques. *IEEE Transactions on Systems Man and Cybernetics. Part B*, **35**(6), 2005. 20

[131] Giorgio Valentini and Thomas G. Dietterich. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. *Journal of Machine Learning Research*, **5**:725–775, 2004. 27, 46, 74

[132] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, **27**(11):1134–1142, 1984. 57

[133] Vladimir N. Vapnik and Alexey Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, **16**(2):264–280, 1971. 55, 57

[134] Ulrike von Luxburg, Olivier Bousquet, and Bernhard Schölkopf. A compression approach to support vector model selection. *Journal of Machine Learning Research*, **5**:293–323, 2004. 4, 56

[135] Geoffrey I. Webb. MultiBoosting: A Technique for Combining Boosting and Wagging. *Machine Learning*, **40**(2):159–196, 2000. 31, 80

[136] Xun-Kai Wei, Ying-Hong Li, Yu-Fei Li, and Dong-Fang Zhang. Enclosing machine learning: concepts and algorithms. *Neural Comput. Appl.*, **17**:237–243, April 2008. 53

[137] D. Wettschereck. A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In Francesco Bergadano and Luc de Raedt, editors, *Proceedings of the European Conference on Machine Learning*, **784** of *LNAI*, pages 323–338. Springer, 1994. 9

[138] D. Wettschereck, D. Aha, and T. Mohri. A review and empirical evaluation of feature-weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review*, **11**(1-5):273–314, 1997. 74

[139] Dietrich Wettschereck and Thomas G. Dietterich. An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. *Machine Learning*, **19**:5, 1995. 9

[140] D. Randall Wilson. Asymptotic properties of nearest neighbor rules using edited data sets. *IEEE Transactions on Systems Man and Cybernetics*, **2**:408–421, 1992. 7

[141] D. Randall Wilson and Tony R. Martinez. Instance pruning techniques. In *Proc. 14th International Conference on Machine Learning*, pages 403–411. Morgan Kaufmann, 1997. 13

[142] D. Randall Wilson and Tony R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, **38**(3):257, 2000. 8, 13, 34, 41

[143] I.H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* 2nd edition Morgan Kaufmann, San Francisco., 2005. 39, 80

[144] Reda Younsi and Anthony Bagnall. An efficient randomized sphere cover classifier. Int. J. of data mining, Modelling and Management, Accepted for publication. 34

[145] Reda Younsi and Anthony Bagnall. A randomized sphere cover classifier. In *The 11th International Conference on Intelligent Data Engineering and Automated Learning (IDEAL)*, pages 234–241, 2010. 34

[146] Zhi-Hua Zhou and Yang Yu. Ensembling local learners through multimodal perturbation. *IEEE Transactions on Systems Man and Cybernetics Part B*, **35**(4):725–735, 2005. 26

Declaration

I herewith declare that I have produced this paper without the prohibited assistance of third parties and without making use of aids other than those specified; notions taken over directly or indirectly from other sources have been identified as such. This paper has not previously been presented in identical or similar form to any other German or foreign examination board.

The thesis work was conducted from XXX to YYY under the supervision of PI at ZZZ.

CITY,