



## An efficient randomised sphere cover classifier

Reda Younsi\* and Anthony Bagnall

School of Computing Sciences,  
University of East Anglia,  
Norwich NR4 7TJ, UK  
E-mail: ry@cmp.uea.ac.uk  
E-mail: ajb@cmp.uea.ac.uk  
\*Corresponding author

**Abstract:** This paper describes an efficient randomised sphere cover classifier ( $\alpha RSC$ ), that reduces the training dataset size without loss of accuracy when compared to nearest neighbour classifiers. The motivation for developing this algorithm is the desire to have a non-deterministic, fast, instance-based classifier that performs well in isolation but is also ideal for use with ensembles. Essentially we trade off decreased testing time for increased training time whilst retaining the simple and intuitive nature of instance-based classifiers. We use 24 benchmark datasets from UCI repository and six gene expression datasets for evaluation. The first set of experiments demonstrate the basic benefits of sphere covering. We show that there is no significant difference in accuracy between the basic  $\alpha RSC$  algorithm and a nearest neighbour classifier, even though  $\alpha RSC$  compresses the data by up to 75%. We then describe a pruning algorithm that removes spheres that contain  $\alpha$  or fewer training instances and compare the results to three data reduction algorithms. The experiments show better average data reduction (89.5%) and improved overall accuracy. The second set of experiments demonstrate that when we set the  $\alpha$  parameter through cross validation, the resulting  $\alpha RSC$  algorithm outperforms several well known classifiers when compared using the Friedman rank sum test. Thirdly, we test the usefulness of  $\alpha RSC$  when used with three attribute filtering methods on six gene expression datasets. Finally, we highlight the benefits of pruning with a bias/variance decomposition. In conclusion, we discuss why the randomisation inherent in  $\alpha RSC$  makes them an ideal ensemble component and outline our future direction.

**Keywords:** classification; sphere cover; machine learning; gene expression classification; bias and variance decomposition; data reduction.

**Reference** to this paper should be made as follows: Younsi, R. and Bagnall, A. (xxxx) 'An efficient randomised sphere cover classifier', *Int. J. Data Mining, Modelling and Management*, Vol. x, No. x, pp.xxx-xxx.

**Biographical notes:** Reda Younsi

---

## 1 Introduction

Instance-based learning techniques (Kibler et al., 1991) operate by keeping a typical sample of the training data then classifying new instances 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 instances, a selection function that selects the samples to be kept by the algorithm, and a classification function that decides on the class of unseen instances. 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 classifiers is that classifying a new instance requires a distance calculation for each instance in the training set. Data reduction algorithms have been studied in great depth (Wilson and Martinez, 2000; Bezdek and Kuncheva, 2000; Kim and Oommen, 2003). In general, these algorithms search the training data for a subset of cases and/or attributes with which to classify new instances with the objective of achieving the maximum compression with the minimum reduction in accuracy. Recently, development in compression scheme rejuvenated research in compression algorithms (Shah, 2008). Compression scheme (Floyd and Warmuth, 1995) is put forward to explain the generalisation performance of sparse algorithms. In general, various algorithms are called sparse because they keep a subset from the training set as part of their learning process (Laviolette et al., 2005). In Younsi and Bagnall (2010), we proposed a new fast and efficient sparse algorithm based on the class cover method which we call randomised sphere cover classifier ( $\alpha RSC$ ). In this paper, we explore the generalisation and compression performance of  $\alpha RSC$  using several experiments. In addition, we test the usefulness of  $\alpha RSC$  on six gene expression datasets. Lastly, we explore the reason behind the good performance of this simple classifier using Domingo’s bias/variance decomposition of the error (Domingos, 2000). The rest of this paper is structured as follows: Section 2 provides some further background into data reduction and class cover problem (CCP); Section 3 describes the  $\alpha RSC$  algorithm and the  $\alpha$  pruning scheme used in the experimentation; The results are reported in Section 5 and finally, we conclude in Section 6 and discuss future direction.

## 2 Background

The sphere covering mechanism we use stems from the class covering approach to classification (Cannon and Cowen, 2000). The CCP involves finding the smallest number of sets covering (i.e., containing) points from one class without covering any points from a second class. The solution to the CCP proposed in Priebe et al. (2003) involves constructing a class cover catch digraph (CCCD), a directed graph based on the proximity of training cases. However, finding the optimal covering via the CCCD is NP-hard (Cannon et al., 2002). Hence (Marchette and Priebe, 2003; Marchette, 2004) proposed a number of greedy algorithms to find an approximately optimal set covering. Whilst covering techniques have shown to be good classifiers that effectively compress the data (Priebe et al., 2003), these algorithms have the drawback that they are still time consuming and that they only find pure coverings, i.e., sets that only contain cases of a single class. An algorithm that relaxes the requirement of class purity was proposed by Priebe et al. (2003). This algorithm introduces two parameters to alleviate this constraint on a pure proper cover. Whilst effective, the parameters are non-intuitive and hard to set. The greedy algorithms proposed all have a run time complexity of  $O(n^2)$ , and hence (to the best of our knowledge) there has been very limited experimental evaluation of the algorithms proposed in Marchette and Priebe (2003) and Priebe et al. (2003) since most of them are impractical for large and complex datasets. The sphere covering algorithm we propose follow the same principles proposed in Marchette et al. (2003). The algorithm is computationally efficient, randomises the process of finding a set covering and allows for pruning through the setting of a single parameter  $\alpha$  which directly penalise complex covers.

## 3 An $\alpha RSC$

The reason for designing the  $\alpha RSC$  algorithm was to develop an instance-based classifier to use in ensembles. Hence our design criteria were that it should be randomised (to allow for diversity), fast (to mitigate against the inevitable overhead of ensembles) and comprehensible (to help produce meaningful interpretations from the models produced). The  $\alpha RSC$  algorithm has a single integer parameter,  $\alpha$ , that specifies the minimum size for any sphere. Informally, for any given  $\alpha$ ,  $\alpha RSC$  works as follows.

- 1 Repeat until all data are covered.
  - a randomly select a data point and add it to the set of covered cases
  - b create a new sphere centred at this point
  - 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  $\alpha$ , 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 1. For all our experiments we use the Euclidean distance metric, although the algorithm can work with any distance function.

---

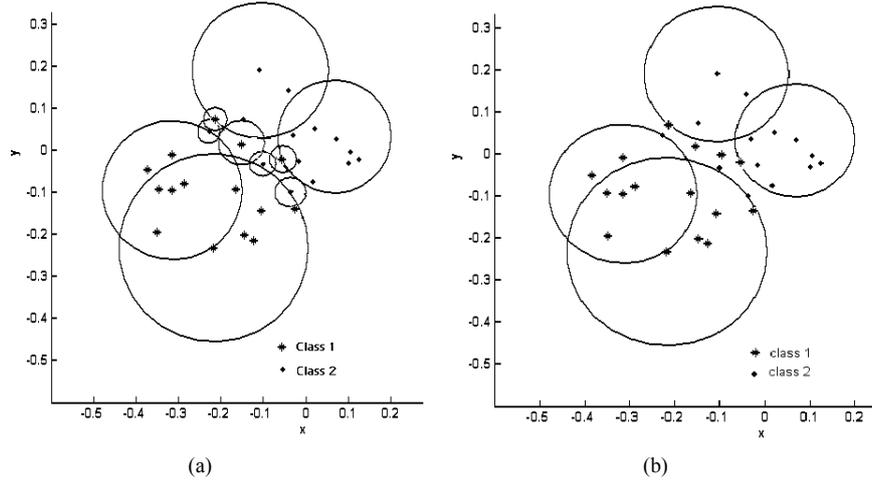
**Algorithm 1** *A  $\alpha$ RSC*


---

- 1: Input: Cases  $D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ , distance function  $d(\mathbf{x}_i, \mathbf{x}_j)$  parameter  $\alpha$ .
  - 2: Output: Set of spheres  $B$
  - 3: Let covered cases set  $C = \emptyset$
  - 4: **while**  $D \neq C$  **do**
  - 5:   Select a random element  $(\mathbf{x}_i, y_i) \in D - C$
  - 6:   Copy  $(\mathbf{x}_i, y_i)$  to  $C$
  - 7:   Find  $\min_{(\mathbf{x}_j, y_j) \in D} d(\mathbf{x}_i, \mathbf{x}_j)$  such that  $y_i \neq y_j$
  - 8:   Let  $r_i = d(\mathbf{x}_i, \mathbf{x}_j)$
  - 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:      $C = C \cup T$
  - 13:     Store the sphere  $B_i$  in  $B$
  - 14:   **end if**
  - 15: **end while**
- 

The parameter  $\alpha$  allows us to smooth the decision boundary, which has been shown to provide better generalisation by mitigating against noise and outliers, (see for instance Liu and Motoda, 2002). Figure 1 provides an example of the smoothing effect of removing small spheres on the decision boundary.

**Figure 1** An example of the smoothing effect of removing small spheres (a) a sphere cover with  $\alpha = 1$  (b) the same cover with  $\alpha = 2$



The  $\alpha RSC$  algorithm classifies a new case by the following rules:

- 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 class covering the test example, the classifier takes the target class of the sphere with the closest centre.
- Rule 2 in the case where a test example is not covered by a sphere, the classifier selects the closest spherical edge.

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 specifically when the high overlap between classes exist [an illustration is given in Figure 1(a)], and areas where noisy cases are within dense areas of examples of different target class. The  $\alpha RSC$  method of compressing through sphere covering and smoothing via boundary setting provides a robust simple classifier that is competitive with other commonly used classifiers.

#### 4 Bias and variance decomposition

In general, the bias/variance theory is used successfully to analyse the generalisation error of any classifier (Valentini and Dietterich, 2004). The main characteristic of the bias and variance decomposition is simplicity of use. The bias/variance decomposition essentially consists in decomposing the generalisation error into two components: bias and variance. In this section, we discuss briefly the bias/variance decomposition for the 0/1 loss function using Domingos framework (Domingos, 2000).

The bias is attributed to the systematic part of the error, while variance to the stochastic part of the error (Domingos, 2000). 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  $t$  training examples  $D_i = \{\mathbf{x}_j, y_j\}_{j=1}^t$ , where each point is a pair  $(\mathbf{x}_j, y_j)$ ,  $y_j \in \mathcal{C}$ ,  $\mathbf{x} \in \mathbb{R}^n$ ,  $n \in \mathbb{N}$ , and  $\mathcal{C}$  is the set of class labels.  $D_i$  can be seen 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 minimising the expected loss  $EL$ . Thus, the expected loss at point  $\mathbf{x}$  can be written as:  $EL(\mathcal{A}, \mathbf{x}) = E_{D_i}[E_y[L(y, f_{D_i}(\mathbf{x}))]]$ ,  $E_{D_i}[\cdot]$  indicate the expected value with respect to the distribution of  $D_i$ .  $E_y[\cdot]$  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) (Domingos, 2000).

*Definition 4.1 [Optimal prediction  $p_*$  (Domingos, 2000)]:* An optimal prediction  $p_*$  is the prediction that the optimal classification algorithm does (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 4.2 [Main prediction  $p_c$  (Domingos, 2000)]:* 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  $B(x)$  is the loss of the main prediction relative to the optimal prediction  $p_*$ .

Bias measures how far are the predictions that a learning algorithm does for an example  $(\mathbf{x}, y)$  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:

$$B(\mathbf{x}) = L(p_*, p_c) \tag{1}$$

*Definition 4.2 [Net variance  $V(\mathbf{x})$  (Domingos, 2000)]:* 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 variance is:

The *net-variance*  $V(\mathbf{x})$  is:

$$V(\mathbf{x}) = E_{D_i}[L(p_c, f_{D_i}(\mathbf{x}))] \tag{2}$$

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.

Domingos decomposition is:

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

Noise part  $c_1N(\mathbf{x})$  is disregarded simplifying the decomposition to

$$EL(\mathcal{A}, \mathbf{x}) = B(x) + c_2V(\mathbf{x})$$

$c_2$  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:

$$\begin{aligned} E_{\mathbf{x}}[EL(\mathcal{A}, \mathbf{x})] &= E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[V_u(\mathbf{x})] - E_{\mathbf{x}}[V_b(\mathbf{x})] \\ &= E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] \end{aligned} \quad (3)$$

To give a simple interpretation, we use a similar illustration presented in Webb (2000). Let  $(\mathbf{x}, y)$ , be an example where  $y \in \mathcal{C} = \{a, b, c\}$  is the target value of an example  $\mathbf{x}$ .

Let say that an algorithm is run ten times on different training set. For each case (example  $(\mathbf{x}, y)$ ), we get a prediction, for a total of ten predictions as shown in Table 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:

$$\text{Case 1} \quad E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] = 0 + ((1 - 0) * 0.6) = 0.6$$

$$\text{Case 2} \quad E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] = 1 + ((1 - 2) * 0.4) = 0.6$$

$$\text{Case 3} \quad E_{\mathbf{x}}[B(\mathbf{x})] + E_{\mathbf{x}}[(1 - 2B(\mathbf{x}))V(\mathbf{x})] = 0 + ((1 - 0) * 0.2) = 0.2$$

**Table 1** Table showing an example of BV calculation

| Correct class   | Case 1 | Case 2 | Case 3 |
|-----------------|--------|--------|--------|
|                 | 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    |

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.

## 5 Results

In this section, we perform four sets of experiments. The first set is meant to demonstrate the general principle that we can intelligently compress the dataset using  $\alpha RSC$  without significantly increasing classification error. The second set of experiments shows that the  $\alpha RSC$  classifier performs as well as or better than other classifiers based on similar principles. The third set of experiments investigates the usefulness of  $\alpha RSC$  in real domains on gene expression datasets. Finally, we show the reason pruning ( $\alpha$  parameter) is the main reason for the good performance of  $\alpha RSC$  using the bias/variance decomposition of the generalisation error.

### 5.1 Experimental setup

To evaluate the performance of  $\alpha RSC$ , we used twenty four datasets from both UCI data repository (Asuncion and Newman, 2007), and boosting repository (<http://ida.first.gmd.de/raetsch/data/benchmarks.htm>). These datasets are summarised in Table 2. 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 2** Benchmark datasets used for the empirical evaluations

| <i>Dataset</i> | <i>Examples</i> | <i>Attributes</i> | <i>Classes</i> | <i>Dataset</i> | <i>Examples</i> | <i>Attributes</i> | <i>Classes</i> |
|----------------|-----------------|-------------------|----------------|----------------|-----------------|-------------------|----------------|
| Sonar          | 208             | 60                | 2              | Vehicle        | 846             | 18                | 4              |
| Glass6         | 214             | 9                 | 6              | Vowel          | 990             | 10                | 11             |
| Glass2         | 214             | 9                 | 2              | German         | 1,000           | 20                | 2              |
| Thyroid        | 215             | 5                 | 2              | Concentric     | 2,000           | 2                 | 2              |
| Heart          | 270             | 13                | 2              | Image segment  | 2,310           | 18                | 2              |
| Haberman       | 306             | 3                 | 2              | Abalone        | 4,177           | 8                 | 3              |
| Cancer         | 315             | 13                | 2              | Clouds         | 5,000           | 2                 | 2              |
| Ecoli          | 336             | 7                 | 8              | Waveforme      | 5,000           | 40                | 3              |
| Ionosphere     | 351             | 34                | 2              | Ringnorm       | 7,400           | 20                | 2              |
| wdbc           | 569             | 30                | 2              | Twonorm        | 7,400           | 20                | 2              |
| Winsconsin     | 699             | 9                 | 2              | Pendigitis     | 10,991          | 14                | 10             |
| Pima diabetes  | 768             | 8                 | 2              | Magic          | 19,020          | 2                 | 10             |

For the experiments in Section 5.2 we used a stratified ten-fold cross validation (CV). In the first experiments, we compare the proposed classifier with several state of the art data reduction algorithms. We used Wilson’s programme written in C for Drop3, IB3, and Explore (Wilson and Martinez, 2000). For the later experiments we performed model selection on the parameter values, in that for each fold of the overall cross-validation we first took a test sample, then cross validated on the remainder to set the parameter. For comparison purposes we compare to K-nearest neighbour (K-NN), the non-nested generalised exemplar (NNge) (Martin, 1995), C4.5, Naive Bayes (NB) and Naive Bayes tree (NBTree). K-NN and NNge are the most relevant for comparison purposes, the other three are included for completeness. Weka implementations are used for the standard classifiers, bespoke implementations for  $\alpha RSC$  and NNge.

**Table 3** 10-fold CV classification accuracy (in %) and standard deviation over the folds

| <i>Dataset</i> | <i>I-NN</i>  | <i>uRSC</i>  | <i>(%)</i> | <i>Dataset</i> | <i>I-NN</i>  | <i>uRSC</i>  | <i>(%)</i> |
|----------------|--------------|--------------|------------|----------------|--------------|--------------|------------|
| 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         |

Note: The final column gives the average compression rate for unpruned RSC (uRSC).

**Table 4** 10-fold CV classification accuracy (in %)

| <i>Dataset</i> | <i>K-NN</i> | <i><math>\alpha</math>RSC</i> | <i>Comp %</i> | <i>IB3</i> | <i>Comp %</i> | <i>Drop3</i> | <i>Comp %</i> | <i>Explore</i> | <i>Comp %</i> |
|----------------|-------------|-------------------------------|---------------|------------|---------------|--------------|---------------|----------------|---------------|
| 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         |

Note: The final column gives the average compression rate (comp) for  $\alpha$ RSC.

### 5.2 Experiment 1: compression without loss of accuracy

The average compression rate for unpruned RSC was 75%. These experiments clearly show that by using the simplest form of  $\alpha$ RSC 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 ( $\alpha$ RSC), we used 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 (Wilson and Martinez, 2000). IB3 is an algorithm which was proposed to rectify shortcomings of the famous IB1 and IB2 (Wilson and Martinez, 2000). Explore was shown to produce very good reduction without too much deterioration in accuracy (Wilson and Martinez, 2000). The results produced in Table 4 shows the best accuracy produced by K-NN and  $\alpha RSC$  trained over a range of parameter values while the reduction algorithms use internal tuning to produce the best results.

The results shown in Table 4 demonstrate once more the good performance of  $\alpha RSC$  in comparison of state of the art data reduction algorithms. In addition, the average accuracy is comparable to that of K-NN.

**Table 5** Classification accuracy (in %) and standard deviation of five classifiers in comparison with  $\alpha RSC$

| <i>Datasets</i> | $\alpha RSC$ | <i>DT</i>    | <i>K-NN</i>  | <i>NB</i>    | <i>NBT</i>   | <i>NNge</i>  |
|-----------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 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 |
| Segment         | 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 |
| F-Avg.          | 3.96         | 2.77         | 5.06         | 3.38         | 3.79         | 2.04         |
| Ranks           | 2            | 5            | 1            | 4            | 3            | 6            |

Notes:  $\alpha RSC$  is used for the baseline. F-Avg stands for Friedman averages and Ranks stands for Friedman ranks

### 5.3 Experiment 2: performance equivalent to other classifiers

The accuracy results in Table 5 are based on an independent test set drawn randomly from the dataset. We use 2/3 of the dataset for training and tested the classifiers on the same remaining test set. However, given the randomisation nature of  $\alpha RSC$ , we choose to use the average of 30 runs on each algorithm in order to make a fair comparison. Tuning the parameters for both  $\alpha$  and K is based on the average of 15 runs using ten CV

on the training set alone. NNge was trained based on the best parameters suggested by its authors. The decision tree is trained without pruning (J48) using the default parameters in WEKA. NB has no parameters.

We are primarily interested in the relative performance of the classifiers over the range of datasets. In order to compare the algorithms on the overall datasets, we use Friedman ranks sum test (Janez, 2006). 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  $N$  datasets. 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 datasets, and it has been shown that the ranking themselves provide a fair comparison of the algorithms (Janez, 2006).  $\alpha RSC$  has the second highest average rank of the five classifiers tested. K-NN has the highest number of top ranks but on some datasets it performed relatively badly. These results suggest that  $\alpha RSC$  can perform well on a variety of datasets in comparison to other classifiers, and that the smoothing process reduces the tendency of  $\alpha RSC$  to perform very badly on some datasets.

#### 5.4 Experiment 3: gene expression datasets

In this section we use six benchmark gene expression datasets in order to evaluate the usefulness on the proposed algorithm in real world application. We know that gene expression datasets contains a very large number of attributes most of which are irrelevant for the classification task (Zhang et al., 2004). For this reason, it was suggested that preprocessing would help to remove unnecessary attributes (Zhang et al., 2004). The two popular preprocessing methods used for the experiments rank best attributes according to the  $\chi^2$  statistics and information gain (IG) (Molina et al., 2002; Guyon and Elisseeff, 2003). In addition, we use the attribute filtering algorithm called relief (RL) (Molina et al., 2002). The three 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 CV (10CV) on the training set only to select the best values for  $\alpha$  and  $K$  based on the average accuracy results of 15 experiments (model selection). The average classification accuracy of 30 experiments is calculated on each test set. For the comparison purpose, we continue with a single Decision tree (J48), NB, K-NN, NBTree and non-nested hyper-rectangle generalisation algorithm (NNge).

The experiments produced 648 accuracy results over the six gene expression datasets using the six classifiers each on three attribute filtering methods ( $\chi^2$ , information gain and RL). In order to collate the results into a single table we calculated the Friedman ranking. Tables 6 shows the best performing classifier for each attribute filtering method. It is interesting to note that  $\alpha RSC$  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  $\alpha RSC$  performed very well over the six gene expression datasets on all the three attribute filtering methods.

**Table 6** The ranking based on classification accuracy of six datasets of  $\alpha RSC$ , K-NN, decision tree (J48), NBTree, NB and NNge using average results of 30 different runs on  $\chi^2$ , IG and RL

| <i>Algorithms</i>           | $\alpha RSC$ | <i>DT</i> | <i>K-NN</i> | <i>NB</i> | <i>NBTree</i> | <i>NNge</i> |
|-----------------------------|--------------|-----------|-------------|-----------|---------------|-------------|
| Ranked all dataset $\chi^2$ |              |           |             |           |               |             |
| Top 5                       | 3            | 5         | 6           | 4         | 1             | 2           |
| Top 10                      | 1            | 6         | 2           | 5         | 3             | 4           |
| Top 20                      | 3            | 6         | 4           | 1         | 2             | 5           |
| Top 30                      | 3            | 6         | 2           | 1         | 5             | 4           |
| Top 40                      | 3            | 6         | 1           | 2         | 4             | 5           |
| Top 50                      | 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       |              |           |             |           |               |             |
| Top 5                       | 2            | 6         | 5           | 4         | 1             | 3           |
| Top 10                      | 2            | 6         | 1           | 4         | 3             | 5           |
| Top 20                      | 3            | 6         | 1           | 4         | 2             | 5           |
| Top 30                      | 5            | 6         | 1           | 2         | 3             | 4           |
| Top 40                      | 4            | 6         | 1           | 5         | 3             | 2           |
| Top 50                      | 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   |              |           |             |           |               |             |
| Top 5                       | 2            | 6         | 4           | 5         | 5             | 3           |
| Top 10                      | 1            | 6         | 3           | 2         | 5             | 4           |
| Top 20                      | 1            | 6         | 3           | 2         | 5             | 4           |
| Top 30                      | 1            | 6         | 3           | 2         | 4             | 5           |
| Top 40                      | 2            | 6         | 3           | 1         | 5             | 4           |
| Top 50                      | 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           |

The best results for each dataset regardless of cut-off points are shown in Tables 7, 8 and 9. 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 mean ranks for each attribute filtering method. The tables show that  $\alpha RSC$  ranked 1st for the  $\chi^2$ , 2nd for RL and 3rd for the Information gain filtering methods.

**Table 7** The best test set accuracy (in %) of  $\alpha RSC$ , K-NN, decision tree (J48), NBTree, NB and NNge using average results of 30 different runs on  $\chi^2$ 

| <i>Dataset</i> | $\alpha RSC$ | <i>NBTree</i> | <i>K-NN</i> | <i>NB</i> | <i>NNge</i> | <i>DT</i> |
|----------------|--------------|---------------|-------------|-----------|-------------|-----------|
| BreastCancer   | 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     |
| Lungcancer     | 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     |
| ColonTumor     | 85.24        | 88.10         | 84.29       | 87.46     | 84.29       | 83.81     |
| CentralNervous | 80.33        | 80.67         | 78.83       | 78.17     | 74.00       | 76.67     |
| Mean rank      | 4.67         | 4.50          | 4.08        | 3.50      | 2.42        | 1.83      |
| Ranks          | 1            | 2             | 3           | 4         | 5           | 6         |

**Table 8** The best test set accuracy (in %) of  $\alpha RSC$ , K-NN, Decision tree (J48), NBTree, NB and NNge using average results of 30 different runs on RL

| <i>Dataset</i> | <i>NB</i> | $\alpha RSC$ | <i>K-NN</i> | <i>NNge</i> | <i>NBTree</i> | <i>DT</i> |
|----------------|-----------|--------------|-------------|-------------|---------------|-----------|
| BreastCancer   | 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     |
| Lungcancer     | 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     |
| ColonTumor     | 85.08     | 86.03        | 80.79       | 82.7        | 82.86         | 79.68     |
| CentralNervous | 78.33     | 77.17        | 76.83       | 70.83       | 70.67         | 71.17     |
| Mean rank      | 4.83      | 4.67         | 4.17        | 2.83        | 2.67          | 1.83      |
| Ranks          | 1         | 2            | 3           | 4           | 5             | 6         |

**Table 9** The best test set accuracy (in %) of  $\alpha RSC$ , K-NN, decision tree (J48), NBTree, NB and NNge using average results of 30 different runs on IG

| <i>Dataset</i> | <i>K-NN</i> | <i>NBTree</i> | $\alpha RSC$ | <i>NNge</i> | <i>NB</i> | <i>DT</i> |
|----------------|-------------|---------------|--------------|-------------|-----------|-----------|
| BreastCancer   | 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     |
| Lungcancer     | 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     |
| ColonTumor     | 85.40       | 86.51         | 85.87        | 84.92       | 84.44     | 82.22     |
| CentralNervous | 77.83       | 82.83         | 74.00        | 75.33       | 75.67     | 74.50     |
| Mean ranks     | 4.50        | 4.33          | 4.00         | 3.08        | 2.92      | 2.17      |
| Ranks          | 1           | 2             | 3            | 4           | 5         | 6         |

The overall ranking results of the three attribute filtering methods is calculated by summing the mean ranks of the three tables as shown in Table 10.  $\alpha RSC$  has ranked first while K-NN ranked 2nd. These results show that  $\alpha RSC$  is a good classifier for these gene expression datasets, and that it works well with attribute filters.

**Table 10** All ranks over the three attribute filtering methods

| <i>Classifiers</i> | <i>Sum ranks</i> | <i>All ranks</i> |
|--------------------|------------------|------------------|
| $\alpha RSC$       | 13.33            | 1                |
| K-NN               | 12.75            | 2                |
| NBTree             | 11.50            | 3                |
| NB                 | 11.25            | 4                |
| NNge               | 8.33             | 5                |
| DT                 | 5.83             | 6                |

### 5.5 Experiment 4: bias/variance decomposition of $\alpha RSC$

For these experiments, we employ ten different datasets, four synthetic (clouds, concentric, twonorm and ringnorm) and the remaining datasets are taken from the UCI repository. 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 datasets we used bootstrapping to replicate the data. In both cases we computed the mean prediction, bias, unbiased and biased variance, net-variance according to the procedures found in Valentini and Dietterich (2004). That is, the data is divided into a training set  $\mathcal{T}_r$  and a test set  $\mathcal{T}_s$ . The training bootstrap samples are much smaller than  $|\mathcal{T}_s|$ . That is,

200 datasets are made from  $\mathcal{T}_r$ , each one consisting of 200 examples uniformly drawn with replacement from  $\mathcal{T}_r$ . Except for two datasets twonorm and waveform, we used 300 training datasets each made of 300 examples.

**Table 11** Comparing best bias and variance results of  $\alpha RSC$  on various datasets

| <i>Dataset</i> | <i>Avg. error</i> | <i>Bias</i>     | <i>Net var.</i> | <i>Var. unb.</i> | <i>Var. bias.</i> |
|----------------|-------------------|-----------------|-----------------|------------------|-------------------|
| 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 %         | ↑ <b>11.01</b>    | ↑ <b>5.32</b>   | ↑ <b>33.81</b>  | ↑ <b>26.78</b>   | ↑ <b>20.80</b>    |
| 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 %         | ↑ <b>17.74</b>    | ↑ <b>9.06</b>   | ↑ <b>38.43</b>  | ↑ <b>31.55</b>   | ↑ <b>21.41</b>    |
| 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 %         | ↑ <b>16.06</b>    | ↓ <b>8.04</b>   | ↑ <b>68.84</b>  | ↑ <b>51.11</b>   | ↑ <b>21.57</b>    |
| 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 %         | ↑ <b>14.70</b>    | ↓ <b>2.89</b>   | ↑ <b>38.44</b>  | ↑ <b>30.78</b>   | ↑ <b>15.91</b>    |
| 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 %         | ↓ <b>2.27</b>     | ↓ <b>125.19</b> | ↑ <b>30.93</b>  | ↑ <b>16.73</b>   | ↓ <b>100</b>      |
| 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 %         | ↑ <b>29.45</b>    | ↑ <b>2.20</b>   | ↑ <b>41.86</b>  | ↑ <b>37.54</b>   | ↑ <b>10.97</b>    |
| 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 %         | ↓ <b>15.92</b>    | ↓ <b>83.66</b>  | ↑ <b>12.35</b>  | ↑ <b>4.50</b>    | ↓ <b>100</b>      |
| 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 %         | ↑ <b>14.30</b>    | ↓ <b>10.62</b>  | ↑ <b>71.67</b>  | ↑ <b>57.16</b>   | ↑ <b>36.73</b>    |
| 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 %         | ↑ <b>3.75</b>     | ↑ <b>0.43</b>   | ↑ <b>15.38</b>  | ↑ <b>10.17</b>   | ↑ <b>10.01</b>    |
| 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 %         | ↑ <b>8.18</b>     | ↓ <b>39.25</b>  | ↑ <b>72.53</b>  | ↑ <b>52.59</b>   | ↑ <b>2.42</b>     |

Notes: 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.

The results in Table 11 show a pattern on the bias and variance performance of  $\alpha RSC$  classifier:

- 1 If pruning improves performance, which it does for the majority of cases in our experiment, there is a substantial decrease in net variance. However, 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 in these datasets, 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 are both decreased.
- 2 If pruning degrades performance, then we notice an increase in bias. This is shown on Pendigitis dataset with 15.92% increase in average error for  $\alpha = 1$ , and a substantial increase in bias (83.66%). Pruning, for this dataset, has also increased substantially the biased variance (100%). Similarly, Concentric dataset 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 dataset is rather unusual; removing spheres that are close to the decision boundary will significantly underfit the data because no separation exist between the two classes. As for Pendigitis dataset, it is made of ten classes which could be an issue for choosing the same  $\alpha$  values for each class. Obviously, for both Concentric and Pendigitis datasets 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 a few cases do we notice a small decrease in bias. Therefore, the decrease of  $\alpha RSC$  average error is caused mainly by the decrease of unbiased variance resulting in the overall error reduction.

## 6 Conclusions

The classification accuracy of our proposed randomised classifier is competitive in comparison to various deterministic algorithms. In addition, the classification accuracy is significantly improved by pruning, to the extent that on average it outranks five other classifiers. The reason for this improvement is the result of unbiased variance reduction as demonstrated by the bias/variance experiments. Pruning is only responsible for the reduction of unbiased variance, which indicates that further improvement is possible by reducing bias. Feature selection is known to reduce bias for IBL classifiers. We intend to investigate the effect of attribute selection on  $\alpha RSC$  and assess the usefulness of the classifier when used in ensembles. Several variations are possible with the sphere cover classifier. For instance, sphere can be allowed to be inside other spheres, and to find the optimum radius of each sphere. Other possibilities might be investigated in regards to sphere positioning. In the future, we will examine the relationships between

$\alpha$ , the accuracy and the cardinality of cover using probabilistic bound based on the compression scheme. Manipulating these three quantities in a compression bound should result in an accurate prediction of the true error, which can be used as an efficient model selection technique. In addition, we will investigate whether further compression is better achieved using kernel methods.

## References

- Asuncion, A. and Newman, D.J. (2007) *Uci. Machine Learning Repository*.
- Bezdek, J.C. and Kuncheva, L. (2000) ‘Some notes on twenty one (21) nearest prototype classifiers’, in *SSPR/SPR*, pp.1–16.
- Cannon, A. and Cowen, L. (2000) ‘Approximation algorithms for the class cover problem’, *AMAI*.
- Cannon, A.J., Ettinger, M., Hush, D. and Scovel, C. (2002) ‘Machine learning with data dependent hypothesis classes’, *The Journal of Machine Learning Research*, Vol. 2, pp.335–358.
- Domingos, P. (2000) ‘A unified bias-variance decomposition for zero-one and squared loss’, in *AAAI/IAAI*, pp.564–569.
- Floyd, S. and Warmuth, M.K. (1995) ‘Sample compression, learnability, and the vapnik-chervonenkis dimension’, *Machine Learning*, Vol. 21, No. 3, pp.269–304.
- Guyon, I. and Elisseeff, A. (2003) ‘An introduction to variable and attribute selection’, *Journal of Machine Learning Research*, Vol. 3, pp.1157–1182.
- Janez, D. (2006) ‘Statistical comparisons of classifiers over multiple data sets’, *The Journal of Machine Learning Research*, Vol. 7, pp.1–30.
- Kibler, D., Aha, D. and Albert, M.K. (1991) ‘Instance-based learning algorithms’, *Machine Learning*, Vol. 6, No. 1, pp.37–36.
- Kim, S-W. and Oommen, B.J. (2003) ‘A brief taxonomy and ranking of creative prototype reduction schemes’, *Pattern Anal. Appl.*, Vol. 6, No. 3, pp.232–244.
- Laviolette, F., Marchand, M. and Shah, M. (2005) ‘Margin-sparsity trade-off for the set covering machine’, in João Gama, Rui Camacho, Pavel Brazdil, Alípio Jorge and Luís Torgo (Ed.): *ECML*, Vol. 3720, pp.206–217, Springer.
- Liu, H. and Motoda, H. (2002) ‘On issues of instance selection’, *Data Mining and Knowledge Discovery*, Vol 6, No. 2, pp.115–130.
- Marchette, D.J. and Priebe, C.E. (2003) ‘Characterizing the scale dimension of a high-dimensional classification problem’, *Pattern Recognition*, Vol. 36, No. 1, pp.45–60.
- Marchette, D.J., Wegman, E.J. and Priebe, C.E. (2003) ‘A fast algorithm for approximating the dominating set of a class cover catch digraph’, Technical report, JHU DMS TR #635.
- Marchette, D.J. (2004) *Random Graphs for Statistical Pattern Recognition*, Wiley-Interscience.
- Martin, B. (1995) ‘Instance-based learning nearest neighbor with generalization’, MSC thesis, University of Waikato, Hamilton, New Zealand.
- Molina, L.C., Belanche, L. and Nebot, À. (2002) ‘Feature selection algorithms: a survey and experimental evaluation’, in *ICDM*, pp.306–313, IEEE Computer Society.
- Priebe, C., DeVinney, J.G., Marchette, D.J. and Socolinsky, D.A. (2003) ‘Classification using class cover catch digraphs’, *Journal of Classification*, Vol. 20, No. 1, pp.3–23.
- Shah, M. (2008) ‘Risk bounds for randomized sample compressed classifiers’, *NIPS*, pp.1449–1456.
- Valentini, G. and Dietterich, T.G. (2004) ‘Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods’, *Journal of Machine Learning Research*, Vol. 5, pp.725–775.
- Webb, G.I. (2000) ‘MultiBoosting: a technique for combining boosting and wagging’, *Machine Learning*, Vol. 40, No. 2, pp.159–196.

- Wilson, R.D. and Martinez, T.R. (2000) 'Reduction techniques for instance-based learning algorithms', *Machine Learning*, Vol. 38, No. 3, p.257.
- Younsi, R. and Bagnall, A. (2010) 'A randomized sphere cover classifier', *Intelligent Data Engineering and Automated Learning – IDEAL*, Lecture Notes in Computer Science, Vol. 6283, pp.234–241.
- Zhang, C., Li, T. and Ogihara, M. (2004) 'Comparative study of attribute selection and multiclass classification methods for tissue classification based on gene expression', *Bioinformatics*, Vol. 20, pp.2429–2437.