# GARF: Towards Self-optimised Random Forests

Mohamed Bader-El-Den and Mohamed Gaber

School of Computing, University of Portsmouth,
Portsmouth PO1 2AW, Hampshire, UK

**Abstract.** Ensemble learning is a machine learning approach that utilises a number of classifiers to contribute via voting to identifying the class label for any unlabelled instances. *Random Forests* RF is an ensemble classification approach that has proved its high accuracy and superiority. However, most of the commonly used selection methods are static. Motivated by the idea of having self-optimised RF capable of dynamical changing the trees in the forest. This study uses a genetic algorithm GA approach to further enhance the accuracy of *RF*. The approach is termed as Genetic Algorithm based *RF* (*GARF*). Our extensive experimental study has proved that *RF* performance is be boosted using the GA approach.

**Keywords:** Random Forest, Genetic Algorithms, Ensemble Classification

## 1 Introduction

Ensemble classification is an established machine learning approach to boost the performance of classification techniques. It is based on the process of building a number of classifiers, and then collectively using them to identify unlabelled instances. Two widely used ensemble approaches could be identified, namely, *boosting* and *bagging*. Boosting is an incremental process of building a sequence of classifiers, where each classifier works on the incorrectly classified instances of the previous one in the sequence. *AdaBoost* [8] is the representative of this class of techniques. However, AdaBoost is pruned to overfitting. The other class of ensemble approaches is the Bootstrap Aggregating (*Bagging*) [5]. Bagging involves building each classifier in the ensemble using a randomly drawn sample of the data, having each classifier giving an equal vote when labelling unlabelled instances. Bagging is known to be more robust than boosting against model overfitting. The main representative of bagging is *Random Forests (RF)* [6]. In *RF* , a number of trees are generated, having each tree built using randomly drawn instances from the data set. Randomisation is also applied when selecting the best node to split on for all the trees. Typically this is an input parameter which is equal to $\sqrt{F}$, where $F$ is the number of features in the data set. More details about *RF* are presented in Section 2.

Genetic algorithm [10]is an optimisation approach that belongs to the family of stochastic optimisation. It has long been used successfully in many applications .The process of applying genetic algorithm goes through four main steps,

initialisation, selection, reproduction and termination. In the initialisation step, an initial population of individual solutions is generated. Using a fitness function, individuals of good performance are used to produce a new generation. This process is the selection step. The reproduction step uses mainly two techniques, crossover and mutation, to produce a new generation. The reproduction process continues until a termination condition is reached.

Motivated by the observation that a number of *RFs* could be drawn from a larger *RF* forming an initial population of individuals, genetic algorithms could be an ideal optimisation solution to build a more accurate ensemble. It is worth noting that this observation also applies to other ensemble approaches. Thus, our hypothesis in conducting this research could be stated as follows: *genetic algorithm is able to further enhance the performance of ensemble classification.*

In this paper, we have proposed, developed and empirically evaluated a novel approach to optimising *RFs* boosting their performance. Our approach is termed Genetic Algorithm based *RFs* (*GARF*). The *GARF* approach starts by generating a large *RF* of $N$ decision trees, forming a vector $\overrightarrow{RF}$. Drawing randomly from $\overrightarrow{RF}$ a number of vectors each denoted as $\overrightarrow{rf_i}$, where the number of trees in $\overrightarrow{rf_i}$ is denoted as $n_i \leq N$, $i = 1..S$, and $S$ is the number of *RFs* . In genetic algorithms terminology, $S$ is the size of the population. This initial population is then evolved through a number of generations, with the fitness function for each individual being its classification accuracy.

The paper is organised as follows. Section 2 provides necessary background about *RFs* and genetic algorithm; the main constructs of our *GARF* technique. Our proposed approach *GARF* to boost the performance of *RFs* is detailed in Section 3. Extensive experimental study validating *GARF* is presented in Section 4. A discussion of related work is given in Section 2.3. Finally, the paper is concluded with a short summary and pointer to future developments in Section 5.

## 2 Background

As we build our *GARF* technique based on *RFs* and genetic algorithm, the following subsections provide necessary background on the two methods.

### 2.1 Random Forests

Two broad categories of techniques could be identified in machine learning, supervised and unsupervised. Supervised techniques are also widely known as classification attempt to identify the value of an attribute, known as the *class attribute*, based on the values of the other attributes in the same instance or record of data. This identification is based on learning from historical data. The attributes other than the class are known as *predictors*. Thus, if the value of the class label is $y$, and the values of the predictors form the vector $\boldsymbol{x}$, then $y = f(\boldsymbol{x})$. Any classification technique attempts to find $\hat{f}(\boldsymbol{x})$ that approximates the function $f(\boldsymbol{x})$.

The notion of using a set of classifiers to identify unlabelled instances is known as ensemble learning. Boosting and bagging are the two known successful approaches to ensemble learning. *RFs* belongs to the bagging approach. *Bagging* (Bootstrap Aggregating) has been proposed by Breiman in [5]. It is based on generating a number of replicas from the training data by uniformly sampling the instances with replacement. This sampling approach is known as *Bootstrap*. It allows duplicate instances to appear in the same replica, and also allows some instances to be left out. Statistically for a large replica that has the number of instances equal to the size of the data set, 63.2% of the instances do appear at least once in the replica. Having a number of replicas, each denoted as $r$ out of the training data, a classifier $c(r)$ is built using the sampled instances in $r$. The classification is done via voting among a vector of classifiers $\overrightarrow{c(r)}$ that have been built using the corresponding vector of replicas $\overrightarrow{r}$. A common performance evaluation approach in bagging is to use out of bag method. This is based on evaluating each instance using those classifiers in the ensemble that did not use that instance for training. This means that not all the classifiers are used together in testing.

Bagging has been applied successfully to an ensemble technique, termed *RFs*. In addition to the Bootstrap sampling, randomisation over the feature space is also used. The technique is based on building a number of decision tree classifiers, having each tree built from one replica out of the training data. However, when splitting the nodes of the decision tree, only a subset of all the features is used. Assuming that the number of features in the data set is $F$, the standard setting for the random features to be used at each split is $M = \sqrt{F}$. Breiman has used *Gini index* as the goodness measure to split the attributes on. *Gini index* has been introduced by Breiman et al [7] in building the Classification And Regression Trees *CART* technique. However, it has been first introduced by the Italian statistician *Corrado Gini* in 1912. The index is a function that could be used to measure the impurity of the data, i.e., how uncertain we are if an event will occur. In classification, this event would be the determination of the class label. The Gini impurity function in its original form is calculated as follows.

$$Gini(t) = 1 - \sum_{i=1}^{w} P(C_i|t)^2 \tag{1}$$

where $t$ is a condition, $w$ the number of classes in the data set, and $C_i$ is the $i^{th}$ class label in the data set.

By removing the condition t from the original form of the previous equation, we can calculate the level of impurity for any data set before splitting as follows.

$$Gini(Class) = 1 - \sum_{i=1}^{w} P(C_i)^2 \tag{2}$$

The *Gini index* of any attribute $A$ can then be calculated as follows.

$$GiniIndex(A) = Gini(Class) - \sum_{j=1}^{m} P(a_j).Gini(A = a_j) \tag{3}$$

where $m$ is the number of values for the attribute $A$.

The attribute with a higher *Gini index* is the one chosen to be split on. It is worth noting that *CART* uses binary splits. Thus, for those attributes with $m > 2$, a preprocessing step is required to find the best was to combine the different values of the attribute to result in a binary split.

### 2.2 Genetic Algorithm

GA is a well established evolutionary approach. Basic details about GA can be found in [9] In an ordinary GA, the chromosome represents an encoded solution. For some problems, the direct encoding of a solution in a GA's chromosome results in complex and large chromosomes that may need complex repairs after the application of the GA's operators. In contrast, [12] introduced what could be called as indirect encoding or *Indirect GAs* (IGAs), where each gene in the chromosome represents a heuristic – this could be seen as rule of thumb, an educated guess or small rules – instead of representing part of the solution. In an *indirect GA*, the chromosome which is known as *Heuristic Chromosome* (HC) may be much more compact and robust, since it represents the heuristics that will be used in order to get a solution.

The most common form of HC, is one where the HC consists of a number of genes and each of these genes represents the ID of a heuristic. In order to build a solution, the heuristics in an HC are called one after the other or in parallel based on the problem and what exactly the chromosome represents. One of the main differences between different HC approaches lies in structure of the HC and what exactly each gene represents.

The approach adopted in this paper is similar to the IGA approach, a single random tree could be considered as a heuristic. Each gene in the chromosome represents a pointer to a random tree classifier, and the chromosome as a whole represents an ensemble classifier (forest). In order to get a solution (classification) of a given instance, the genes in the chromosome are used to evaluate the instance as detailed in section 3.

### 2.3 Related Work

Genetic algorithm has been applied in machine learning and data mining extensively. The main application is the use of genetic algorithm in the feature selection problem. An early survey on this topic can be found in [11]. However, the relevant work to the research reported in this paper is detailed in the following.

Robnik-Sikonja [13] has proposed possible extensions to *RFs* that have proved to boost the accuracy of the original techniques presented in Section 2. The motivation behind these extensions is to decrease the correlation among the trees in the *RF*. As the original technique proposed by [6] uses *Gini index* for finding the best split among the randomised vector of attributes $\overrightarrow{F_M}$. In an attempt to decrease the dependency among attributes, Robnik-Sikonja has used ReliefF

[14] as a measure of the quality of the attributes. This extension has not proved to have a good performance on real data sets. A combination of measures for the quality of attributes has been used to decide the split, having each fifth tree in the forest uses a different measure. This method has proved to boost the performance of the *RF*, but not significantly. The other approach proposed by Robnik-Sikonja was the use of weighted voting among the trees using similarity of the instances with regards to their performance on the individual trees. This method always has proven to boost the performance of the input forest, and very competitive with other state-of-the-art methods.

Sylvester and Chawla [15] have proposed the *EVEN* (EVolutionary ENsembles). They have also attempted to use weighted voting among a set of homogeneous or heterogeneous classifiers. The *EVEN* system uses each of the classifier's performance over a validation set of data to weight the tree. Experimental validation has proved that *EVEN* can outperform the unweighted ensemble. In a more recent work, Abdulsalam and Skillicorn [2] have used Hoeffding trees to build a window of *RFs* to tackle the concept drift problem when mining streaming data.

## 3 GARF

*GARF* uses variable size chromosomes. Each chromosome (individual) in the population represents a forest. Each of the genes in the chromosome represents a random tree. Traditional genetic operators are employed by the proposed GARF for the crossover; a standard single point crossover operator is adopted. Two modes of operation for the crossover operator have been developed and tested. In the first mode all the repeated genes that could occur because of the crossover in the new individuals are removed. This to make sure that the each evolved forest has no repeated trees. The second mode does not make this extra check and allows the repetition of the trees in the offspring. For the mutation, a standard uniform mutation operator is employed, where the operator replaces a randomly chosen tree/gene with another randomly selected tree from the input trees forest that does not already exist in the forest/individual.

Each dataset is divided into three sets, training, validation (for GA training) and testing. The training set is used for building the random tress (input RF). The accuracy of the trees during the training is very high, reaching in most cases above 99% accuracy. By the accuracy here we mean the ability of correctly classifying a given instance. This is because these instances have been seen before during the building stage and in random trees does not use burning. As a result, it is not possible to use these instances (training set) for training the GARF as well, and another indebtedness set of instances (Optimization set) is needed for training the GA. In this paper we may refer to the validation set as GA-training set.

### 3.1 Fitness

Before GRAF starts the evolution process, each tree in the input forest is used to classify each of the instances in the GA-training set, all the classification

results of every tree for each instance is stored in a buffer. This is done to speed up the evolution process and especially the fitness evaluation. So in the fitness evaluation of each individual, instead of evaluating the performance of all the trees in the individual against all the instances in the GA-training set, the classification results are collected directly from the buffer.

A given instance is considered as correctly classified, if the number of trees in the individual that has correctly classified it is greater than the number of trees that have given incorrect classification. In contrast, a given instance is considered as incorrectly classified, if the number of trees in the individual that has correctly classified it is less than or equal to the number of trees that have given incorrect classification. We call it a tie, if the number of trees that has correctly classified the instance is equal to the number of the instances that have been incorrectly classified,

The fitness of the individual is based on the number instances he has correctly classified.

$$f(v) = \sum_{i}^{K} c(v,i) + \frac{s(v,i)}{K} \tag{4}$$

where $K$ is the number of instances in the validation set. $c(v,i)$ return 1 if individual $v$ has correctly classified instance number $i$ 0 otherwise. $s(v,i)$ return 1 if it is a tie 0 otherwise.

If it is a tie we consider it as an incorrect classification. However, this could mean that the performance of the individual could be improved by a small change in the trees combination, and may benefit more from the genetic operators. Therefore, we slightly increase the fitness of the individual by $1/K$ for each tie.

### 3.2 GARF Algorithm

In this section, we provide details of our *GARF* method in an algorithmic format. The algorithm is depicted in Algorithm 1, where $NG$ is the number of generations in GA, $S$ denoting the size of the population (number of individual random forests) and $n$ is the size of individual random forests in the initial population.

Having presented our proposed *GARF* technique in detail, the following section has validated the technique via extensive experimental study.

## 4 Experimental Study

We have conducted a series of experiments to evaluate the performance of *GARF* against the state of the art classification techniques. For our experiments, we used Waikato Environment for Knowledge Analysis *(WEKA)* [16]. We compared the performance of *GARF* against state of the art classification techniques; C4.5 decision tree, Support Vector Machines *(SVM)* and AdaBoost. We have also used WEKA to build the *RF*, we denote this as *RFweka*. The initial *RF* on which we used to build our initial population of *RFs* has been built using single calls of the random tree technique in WEKA. We denoted this in our experiments

**Algorithm 1** GARF Algorithm

{User Settings}
input $N$, $M$, $S$, $NG$
{Process}
$\overrightarrow{RF}$ = Call RandomForest($N$, $M$)
**for** $i = 1 \rightarrow S$ **do**
  **for** $k = 1 \rightarrow n$ **do**
    $x = Random(1 \rightarrow N)$
    Add tree $RF_x$ to forest $i$ in the GA population $\overrightarrow{P_i}$
  **end for**
**end for**
Evaluate each forest in the initial population $\overrightarrow{P}$
**for** $j = 1 \rightarrow NG$ **do**
  {Generate a new population by applying GA: operators mutation and crossover}
  $\overrightarrow{PNew} = GAOperators(\overrightarrow{P})$
  Evaluate each forest in $\overrightarrow{P}$
  $\overrightarrow{bestForest} \leftarrow$ copy of best $\overrightarrow{P}$
  $\overrightarrow{P} = \overrightarrow{PNew}$
**end for**
{Output}
A vector of trees $\overrightarrow{bestForest}$

as *RFin*. *RFin* was not created as one forest. Instead WEKA was used to create *RFin* as a set of independent random trees to enable us to evaluate each tree separately.

We have used 15 real standard data sets from UCI repository [1]. Description of the used data sets is given in Table 2. We have used a variety of data sets with diversity in the number of instances, number of classes and number of attributes.

As aforementioned, we have divided the data sets into three equal parts; one third for training, one third for optimisation (validation), and one third for testing. In *GARF*, we have used the validation part to evolve our *RFs* . To conduct fair experiments, we have combined the training and validation parts of the data sets to be used for training the other techniques. The same testing set has been used to calculate the performance of all the used classifiers.

The results of the experiments are shown in Table 2. The table shows our *GARF* technique has always been superior than the initial *RF (RFin)*. The target in this paper is improving the performance of any given RF by changing the trees in the forest, which has been achieved. However, we compared the performance of *GARF* with state-of-the-art tree and bagging classifiers. Out of the 15 data sets, *GARF* has performed the best over all the other classification techniques in 8 data sets. For the letter data set, WEKA has not been able to scale to run such a large data set for the classifiers we have used. But as shown in the table, *GARF* has outperformed the initial *RF*. The *GARF* results shown in Table 2 are the 10 runs, parameters settings and statistics of these results are shown in Table 3 under Experiment 1.

Table 1: Data Sets Used

| Name | Ins | Class. | Train. | Valid. | Test. | Name | Ins | Class. | Train. | Valid. | Test. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| diabetes | 768 | 2 | 256 | 256 | 256 | glass | 214 | 7 | 71 | 72 | 71 |
| ionosphere | 351 | 2 | 118 | 117 | 116 | iris | 150 | 3 | 50 | 50 | 50 |
| labor | 57 | 2 | 20 | 19 | 18 | soybean | 683 | 19 | 228 | 228 | 227 |
| vote | 435 | 2 | 145 | 145 | 145 | credit-g | 1000 | 2 | 333 | 334 | 333 |
| ecoli | 336 | 8 | 110 | 113 | 113 | letter | 20000 | 17 | 6666 | 6667 | 6667 |
| liver-disorders | 345 | 7 | 113 | 116 | 116 | sonar | 208 | 61 | 69 | 70 | 69 |
| vehicle | 846 | 19 | 282 | 282 | 282 | vowel | 990 | 14 | 330 | 330 | 330 |
| waveform-500 | 5000 | 41 | 1665 | 1668 | 1667 | | | | | | |

Table 2: Performance of *GARF* against state-of-the-art techniques

| Data Set Name | GARF | RFin | RFweka | AdaBoost | C4.5 | SVM |
|---|---|---|---|---|---|---|
| diabetes | 78.5156 | 76.5625 | 75.3906 | **80.0781** | 76.5625 | 79.2969 |
| glass | 71.8310 | 67.6056 | **76.0563** | 33.8028 | 59.1549 | 47.8873 |
| ionosphere | **95.6896** | 93.1034 | 92.2414 | 91.3793 | 93.1034 | 91.3793 |
| iris | **96.0000** | 92.0000 | 94.0000 | **96.0000** | **96.0000** | 90.0000 |
| labor | **94.4444** | 88.8889 | 77.7778 | 83.3333 | 77.7778 | 83.3333 |
| soybean | 85.4626 | 81.4978 | **87.2247** | 32.5991 | 83.7004 | N/A |
| vote | 96.5517 | 95.8621 | 98.6207 | **99.3103** | 97.2414 | 7.2414 |
| credit-g | **73.8739** | 72.3724 | 72.6727 | 68.4685 | 69.6697 | 71.4715 |
| ecoli | **71.6814** | 69.9115 | 69.0265 | 24.7788 | 68.1416 | 61.0619 |
| letter | **84.0108** | 83.3508 | N/A | N/A | N/A | N/A |
| liver-disorders | **69.8276** | 65.5172 | 68.9655 | 59.4828 | 61.2069 | 57.7586 |
| sonar | **88.4058** | 85.5072 | 81.1594 | 76.8116 | 76.8116 | 84.058 |
| vehicle | 73.7589 | 70.9220 | **74.8227** | 38.6525 | 65.9574 | 66.3121 |
| vowel | 74.5455 | 73.0303 | **80.0000** | 15.7576 | 65.1515 | 51.5152 |
| waveform-500 | 85.1830 | 84.5231 | 85.0030 | 73.0054 | 74.1452 | **86.0828** |

To assess the robustness of *GARF* with varying the experimental settings of genetic algorithm, we have conducted a set of experiments with different settings. The corresponding results are presented in Table 3. It can be noted that *GARF* has proved to be robust with the various setting of parameters, achieving consistently good accuracy over all the data sets used in our experimental study.

The above results open the door for a range of possibilities to build on the success of *GARF*. One important success factor for any ensemble of classification is to increase the diversity among the classifiers. We note that throughout our experimental study we have used a fixed number of randomised features $M = \sqrt{F}$. However, having a variable $M$ will lead to having a more diverse trees in the *RF*. Evolving this diversity of trees using genetic algorithm would have the potential to further improve *GARF*.

Moreover, given the empirically validated robustness of *RFs* against noise, it is suitable to address the problem of changing data, known as concept drift. *GARF* can address this issue, because of the natural evolution of genetic algorithm. However, an important issue needs to be addressed. One one hand,

Table 3: Performance of *GARF* with Varying Experimental Settings.
Experiment 1: PSize 100, NG 50, CR 0.9, MR 0.1 and variable InvSize of 100.
Experiment 2: PSize 100, NG 50, CR 0.9, MR 0.1 and fixed InvSize of 200.
Experiment 3: PSize 500, NG 50, CR 0.9, MR 0.1 and variable InvSize of 100.
Experiment 4: PSize 400, NG 50, CR 0.9, MR 0.1 and variable InvSize of 400.
where PSize = Population Size, NG = Number of Generations, CR = crossover rate, MR = mutation rate, InvSize = GA string length (individual).

| | Experiment 1 | | | | Experiment 2 | | | |
|---|---|---|---|---|---|---|---|---|
| Data Set Name | Best | Size | Worst | Std. | Best | Size | Worst | Std. |
| | Best | Size | Worst | Std. | Best | Size | Worst | Std. |
| diabetes | 78.5156 | 89 | 76.5625 | 3.124 | 77.7344 | 200 | 75.7813 | 3.1162 |
| glass | 71.831 | 87 | 67.6056 | 2.8169 | 73.2394 | 200 | 69.0141 | 2.9001 |
| ionosphere | **95.6896** | 87 | 93.1034 | 3.8204 | 94.8276 | 200 | 93.1034 | 3.7099 |
| iris | 96 | 95 | 92 | 3.6878 | 96 | 200 | 92 | 3.9396 |
| labor | 94.4444 | 95 | 88.8889 | 2.2103 | 94.4444 | 200 | 88.8889 | 2.7011 |
| soybean | 85.4626 | 89 | 82.8194 | 3.4327 | 84.141 | 200 | 81.9383 | 3.3849 |
| vote | 96.5517 | 91 | 95.1724 | 3.8448 | 96.5517 | 200 | 95.1724 | 3.8399 |
| credit-g | 73.8739 | 97 | 71.7718 | 2.974 | 73.5736 | 200 | 71.4715 | 2.9558 |
| ecoli | 71.6814 | 89 | 64.6018 | 3.0656 | 70.7965 | 200 | 68.1416 | 2.7534 |
| letter | 84.0108 | 89 | 83.5458 | 3.3576 | 83.7708 | 200 | 83.3358 | 3.3423 |
| liver-disorders | 69.8276 | 89 | 66.3793 | 2.8172 | 68.9655 | 200 | 65.5172 | 2.7801 |
| sonar | 88.4058 | 95 | 84.058 | 3.4174 | 88.4058 | 200 | 84.058 | 3.5381 |
| vehicle | 73.7589 | 89 | 70.922 | 2.9694 | 73.7589 | 200 | 71.6312 | 2.9846 |
| vowel | 74.5455 | 89 | 72.7273 | 2.9536 | 74.2424 | 200 | 72.7273 | 2.9709 |
| waveform-500 | 85.183 | 87 | 83.9232 | 3.3813 | 84.823 | 200 | 84.0432 | 3.3816 |
| | Experiment 3 | | | | Experiment 4 | | | |
| Data Set Name | Best | Size | Worst | Std. | Best | Size | Worst | Std. |
| diabetes | 78.5156 | 67 | 76.1719 | 3.123 | 78.125 | 154 | 76.1719 | 3.124 |
| glass | 70.4225 | 71 | 64.7887 | 2.8727 | 70.4225 | 171 | 67.6056 | 2.6575 |
| ionosphere | 96.5517 | 69 | 93.1034 | 3.8476 | 95.6897 | 131 | 93.1034 | 3.7853 |
| iris | 96 | 69 | 92 | 3.7094 | 96 | 161 | 92 | 3.7524 |
| labor | 94.4444 | 71 | 88.8889 | 2.3012 | 94.4444 | 161 | 88.8889 | 2.4125 |
| soybean | 88.5463 | 67 | 85.022 | 3.5571 | 85.9031 | 169 | 83.7004 | 3.4316 |
| vote | 96.5517 | 71 | 95.1724 | 3.8621 | 97.2414 | 167 | 95.1724 | 3.82 |
| credit-g | 74.7748 | 72 | 71.1712 | 3.0263 | 73.8739 | 161 | 71.7718 | 3.0042 |
| ecoli | 69.0265 | 73 | 63.7168 | 2.8263 | 70.7965 | 161 | 66.3717 | 2.9458 |
| letter | 84.3558 | 65 | 83.6508 | 3.3602 | 83.9658 | 149 | 83.5908 | 3.3521 |
| liver-disorders | 69.8276 | 71 | 66.3793 | 2.8799 | 69.8276 | 167 | 66.3793 | 2.7694 |
| sonar | 88.4058 | 69 | 81.1594 | 3.6435 | 88.4058 | 131 | 84.058 | 3.3803 |
| vehicle | 73.7589 | 67 | 70.922 | 2.9165 | 73.7589 | 169 | 71.6312 | 2.9559 |
| vowel | 75.1515 | 69 | 72.4242 | 3.0454 | 74.5455 | 151 | 72.7273 | 2.9654 |
| waveform-500 | 85.4229 | 71 | 83.8632 | 3.4368 | 85.243 | 161 | 83.9232 | 3.4025 |

sudden and strong concept drift requires new trees to be added to the forest. On the other hand, gradual and weak concept drift can easily utilise the genetic algorithm to use existing trees in the *RF*. Extensions to *GARF* using more pow-

erful powerful methods such as Genetic Programming [4][3] to address this issue would also have a great potential in the data stream mining area, also with the use of more .

## 5 Conclusion

In this paper, we have empirically validated our novel approach to developing an optimised *Random Forest (RF)* using genetic algorithms that we termed *GARF*. The approach is based on generating a large *RF*, which is decomposed into a number of smaller *RF* . The smaller forests are composed of trees drawn randomly with replacement from the initial large *RF*. Genetic algorithm is an optimisation technique which is then applied to evolve this initial population of individual *RF* with the fitness function being the classification of the forest.

## References

1. D. N. A. Asuncion. UCI machine learning repository, 2007.
2. H. Abdulsalam, D. B. Skillicorn, and P. Martin. Classification using streaming random forests. *IEEE Trans. Knowl. Data Eng.*, 23(1):22–36, 2011.
3. M. Bader-El-Den and R. Poli. Generating SAT local-search heuristics using a GP hyper-heuristic framework. In *Evolution Artificielle, 8th International Conference*, volume 4926, pages 37–49, Tours, France, 29-31 Oct. 2007. Springer.
4. M. B. Bader-El-Den, R. Poli, and S. Fatima. Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. *Memetic Computing Journal*, 1(3):205–219, 2009.
5. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
6. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
7. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.
8. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1995.
9. R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. Wiley, 2004.
10. J. H. Holland. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, USA, 1992.
11. M. Martin-Bautista and M.-A. Vila. A survey of genetic feature selection in mining issues. In *Proceedings of the Congress on Evolutionary Computation. CEC 99.*, volume 2, 1999.
12. I. Norenkov. Scheduling and allocation for simulation and synthesis of cad system hardware. In *In Proceedings EWITD 94, East-West International Conference, ICSTI*, pages 20–24, Moscow, 1994.
13. M. Robnik-Sikonja. Improving random forests. In *ECML*, pages 359–370, 2004.
14. M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn.*, 53:23–69, October 2003.
15. J. Sylvester and N. Chawla. Evolutionary ensemble creation and thinning. In *International Conference on Neural Networks.*, pages 5148–5155. IEEE, 2006.
16. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition, 2005.