Optimisation of Ensemble Classifiers using Genetic Algorithm

Mohamed Medhat Gaber and Mohamed Bader-El-Den

School of Computing, University of Portsmouth,
Portsmouth PO1 3HE, Hampshire, UK
{Mohamed.Gaber,Mohamed.Bader}@port.ac.uk

Abstract. Ensemble learning is a well established machine learning approach that utilises a number of classifiers to aggregate the decision about determining the class label. In its basic form this aggregation is achieved via majority voting. A generic approach, termed *EV-Ensemble*, for evolving a new ensemble from an existing one is proposed in this paper. This approach is applied to the high performance ensemble technique *Random Forests*. This study uses a genetic algorithm approach to further enhance the accuracy of *Random Forests*, based on the *EV-Ensemble* approach. The new technique is termed as Genetic Algorithm based Random Forests (*GARF*). Our extensive experimental study has proved that Random Forests performance could be boosted when evolved using the genetic algorithm 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 use them all 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 [5] 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*) [2]. 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* [3]. In random forests, a number of trees are generated, having each tree built using a 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 random forests are presented in Section 3.

In this paper, we propose to evolve an initial ensemble of classifiers to boost its performance. We termed this generic approach as *EV-Ensemble*, having the first two characters referring to the evolutionary approach of building the ensemble. *EV-Ensemble* is based on the theory that we can always derive an ensemble from a larger one that has at least the same accuracy.

Genetic algorithm [7]is an optimisation approach that belongs to the family of stochastic optimisation. It has long been used successfully in many applications [6]. 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 random forests could be drawn from a larger random forest 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 random forests boosting their performance. Our approach is termed Genetic Algorithm based Random Forests (*GARF*). The *GARF* approach starts by generating a large random forest 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 $\overrightarrow{rf_i}$ denoted as $n_i \leq N$, $i = 1..S$, and S in the number of random forests. 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. A discussion of related work is given in Section 2. Section 3 provides necessary background about ensemble learning, random forests and genetic algorithm. The theoretical underpinnings of our proposed approach *EV-Ensemble* are presented in section 4. Our proposed approach *GARF* to boost the performance of random forests is detailed in Section 5. Extensive experimental study validating *GARF* is presented in Section 6. Finally, the paper is concluded with a short summary and pointer to future developments in Section 7.

2 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 could be found in [8]. However, the relevant work to the research reported in this paper is detailed in the following.

Robnik-Sikonja [11] has proposed possible extensions to random forests that have proved to boost the accuracy of the original techniques presented in Section 3. The motivation behind these extensions is to decrease the correlation among the trees in the random forest. As the original technique proposed by [3] 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 [12] as a measure of the quality of the attributes. This extension has not proved to have a good peformance 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 random forest, 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 has proven to always boost the performance, or at least being as good as the performance of the original the random forests technique over a number of real data sets.

Sylvester and Chawla [13] 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.

3 Background

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

3.1 Ensemble Classification and Random Forests

Two broad categories of techniques could be identified in machine learning, supervised and unsupervised. Classification techniques belong to the supervised learning category, in which a classifier induced from the data attempts 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. Random forests belongs to the bagging approach. *Bagging* (Bootstrap Aggregating) has been proposed by Breiman in [2]. It is based on generating a number 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 [2]. 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 *Random Forests*. 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 Brieman et al [4] 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 about an event to 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. The *Random Forests* algorithm is depicted in Algorithm 1, having N be the number of trees in the random forest, F be the total number of features in the data set, M be the number of features to split on at each node, $\overrightarrow{F_M}$ be the vector of M features to split on, T_i be the i^{th} tree in the random forest, $B(\overrightarrow{F_M})$ be the best feature to split on, and $\overrightarrow{RF}$ be the vector of all trees in the random forest.

It has been also established empirically that setting the number of trees N in the forest to 100 or more will yield the best results [15]. However, increasing N beyond 100 mostly will not have much effect on the accuracy negatively.

Algorithm 1 Random Forests Algorithm

```
{User Settings}
input $N$, $M$
{Process}
Create an empty vector $\overrightarrow{RF}$
for $i = 1 \to N$ do
  Create an empty tree $T_i$
  repeat
    Sample $M$ out of all features $F$ using Bootstrap sampling
    Create a vector of the $M$ features $\overrightarrow{F_M}$
    Find Best Split Feature $B(\overrightarrow{F_M})$
    Create A New Node using $B(\overrightarrow{F_M})$ in $T_i$
  until No More Instances To Split On
  Add $T_i$ to the $\overrightarrow{RF}$
end for
{Output}
A vector of trees $\overrightarrow{RF}$
```

3.2 Genetic Algorithm

GA is a well established evolutionary approach. Basic details about GA could be found in [6]. 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, in [9] the authors have introduced what could be called as indirect encoding or *Indirect GAs* (IGAs), where each gene in the chromosome represents a heuristic – 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 may be much more compact and robust, since it represents the heuristics that will be used in order to get a solution. A chromosome that represents heuristics instead of a is know as a *Heuristic Chromosome* (HC).

The most common form of HC, whether, 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 the IGA [9] approach, a single random tree could be considered as a heuristic. Each gene in the chromosome represent a pointer to a random tree classifier, and the chromosome as a whole represent 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 5.

4 EV-Ensemble

EV-Ensemble is our novel approach to building an ensemble of classification. It is based on the following theorem.

Theorem 1. *Having a set of classifiers C that form an ensemble E with a number of classifiers $|C| = N$, $\exists\ C' \subseteq C$ forming an ensemble $\hat{E}$ with a number of classifiers $|C'| = n \mid Accuracy(\hat{E}) \geq Accuracy(E)$ and $n \leq N$.*

Proof. If all subsets of C denoted as $\Psi(C)$, at least a single subset $C^{\dagger} \in \Psi(C)$ forming the $\hat{E}$ will have the maximum accuracy of all ensembles formed using individual subsets of $\Psi(C)$, given that $C \subseteq C$. Thus in the worst case, if no other subset has proved a greater accuracy, C is chosen as the subset to form $\hat{E}$.

In Theorem 1, it is stated that for an ensemble with a large number classifiers, it is possible to find another ensemble formed from this large pool of classifiers such that its performance is at least as good as the initially formed ensemble. This theorem always holds given that it is possible to use the same superset of classifiers to build the ensemble, if any of its proper subsets except $\emptyset$ proved not to boost the performance of the ensemble.

Based on Theorem 1, we can derive the following corollary that forms the base of our *GARF* technique.

Corollary 1. *Given $C'_i \subseteq C$ and $i = 1 \ldots t$, with some probability δ, it is possible to evolve an ensemble $\hat{E}$ from all the C'_i, such that $Accuracy(\hat{E}) \geq Accuracy(E)$*

In Corollary 1, we state that if we can create a number of ensembles, each drawn randomly from the pool of classifiers that form an initial large ensemble, with some probability δ we can create an ensemble $\hat{E}$ that has a higher accuracy than the initial ensemble. The δ can greatly increase if a well developed optimisation method is used, however, the calculation of δ is out of the scope of this work. In this work, we have used genetic algorithms, because of its empirically proven performance. Our work basically aims at empirical proof of Corollary 1.

Having briefly discussed the theoretical underpinnings of our approach, in the following sections, we discuss our novel technique *GARF* that evolves a new ensemble from an initial random forest and experimentally proves its superiority not only over the initial random forest, but also when compared with state-of-the-art techniques.

5 GARF

GARF uses variable size chromosome. Each chromosome (individual) in the population represents a forest. Each of the genes in the chromosome represents a random tree. Traditional genetic operators are used by the proposed *GARF*. For the crossover, a standard single point crossover operators is adopted, where a single crossover point on both parent's individuals strings is selected. All data beyond that point in both parents are swapped resulting in two new individuals. For the mutation, a standard uniform mutation operator is employed, 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 random forest). 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 independent set of instances (optimisation set) is used for training the GA. In this paper we refer to the validation set as EV-training set.

5.1 Fitness

Before *GRAF* starts the evolution process, each tree in the input forest is used to classify each of the instances in the EV-training set, all the classification results for all the trees on all the instances are stored in a buffer. This is done to speed up the evolution process and especially the fitness evaluation. Consequently, 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 EV-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 grater than the number of trees that has 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 the number of trees that has 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 trees that has 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, and 0 otherwise. $s(v,i)$ return 1 if it is a tie and 0 otherwise.

If it is a tie, we consider it as an incorrect classification. However, this may indicate that the performance of the individual could be improved by 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.

5.2 GARF Algorithm

In this section, we provide details of our *GARF* method in an algorithmic format. Using the same notation introduced in Section 3, with the addition of NG representing the number of generations in genetic algorithm, 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, Algorithm 2 describes the proposed procedure.

Algorithm 2 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 $\overrightarrow{bestForest}$
```

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

6 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 the state of the art classification techniques; C4.5 decision tree [10], Support Vector Machines *(SVM)* [14] and AdaBoost [5], which is a competitor ensemble classification. We have used also WEKA to build the random forest, we denote this as *RFweka.* The initial random forest that we used to build our initial population of random forests has been built using single calls of the random tree technique in WEKA. We denoted this in our experiments 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]. 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 random forests. 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 1. The table shows our *GARF* technique has been always superior than the initial random forest *(RFin)*. More importantly, 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 random forest.

Table 1: 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	**96.5517**	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**

7 Conclusion and Future Work

In this paper, we have empirically validated our novel approach to developing an optimised random forest using genetic algorithms that we termed *GARF*. *GARF* is based on our theoretical framework for evolving a large ensemble of classifiers that we termed *EV-Ensemble*. The approach, when applied in *GARF*, is based on generating a large random forest, which is decomposed into a number of smaller random forests. The smaller forests are composed of trees drawn randomly with replacement from the initial large random forest. Genetic algorithm is an optimisation technique is then applied to evolve this initial population of individual random forests with the fitness function being the classification of the forest. Thanks to the proven success of *GARF*, the application of genetic algorithms to other forms of ensemble classification will be our next step in this research. We also have a long-term plan to use the *EV-Ensemble* approach with heterogeneous and homogeneous ensembles, utilising other optimisation techniques.

References

1. D. N. A. Asuncion. UCI machine learning repository, 2007.
2. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
3. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
4. 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.
5. Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting, 1995.
6. R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms with CD-ROM.* Wiley-Interscience, 2004.
7. J. H. Holland. *Adaptation in natural and artificial systems.* MIT Press, Cambridge, MA, USA, 1992.
8. M. Martin-Bautista and M.-A. Vila. A survey of genetic feature selection in mining issues. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 2, pages 3 vol. (xxxvii+2348), 1999.
9. 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.
10. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
11. M. Robnik-Şikonja. Improving random forests. In *ECML*, pages 359–370, 2004.
12. M. Robnik-Šikonja and I. Kononenko. Theoretical and empirical analysis of relieff and rrelieff. *Mach. Learn.*, 53:23–69, October 2003.
13. J. Sylvester and N. Chawla. Evolutionary ensemble creation and thinning. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pages 5148–5155. IEEE, 2006.
14. V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.
15. G. Williams. Data mining: Desktop survival guide. Togaware, 2010.
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.