# A genetic approach for training diverse classifier ensembles

**David Gacquer, François Delmotte, Veronique Delcroix, Sylvain Piechowiak**
Laboratoire d'Automatique, de Mécanique et d'Informatique, Industrielles et Humaines
Université de Valenciennes et du Hainaut-Cambrésis - France
david.gacquer@univ-valenciennes.fr

## Abstract

Classification is an active topic of Machine Learning. The most recent achievements in this domain suggest using ensembles of learners instead of a single classifier to improve classification accuracy. Comparisons between Bagging and Boosting show that classifier ensembles perform better when their members exhibit diversity, that is commit different errors. This paper proposes a genetic algorithm to design classifier ensembles, using a fitness function based on both accuracy and diversity. The proposed implementation has been run on several UCI Machine Learning datasets and compared to the performances obtained with bagging algorithm and a single classifier of the same nature.
**Keywords:** Classification, Classifier Ensembles, Diversity.

## 1 Introduction

Supervised classification is an active area of research in the Machine Learning Theory. Consider a dataset $t = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ where each $x_i \in X \subset \mathbb{R}^n$ is an input vector and each $y_i \in \Omega = (w_1, w_2, \ldots, w_c)$ is a class label. The objective is to learn the target function $f : \mathbb{R}^n \rightarrow \Omega$ from $t$ to map new unlabeled input vectors to their corresponding class. The Machine Learning literature is extremely large regarding pattern recognition and classification [5, 15]. Recent achievements in the field of supervised learning suggest that combining different learners instead of a single classifier can improve accuracy [10]. Well known illustrations of this theory are bagging and boosting algorithms [7, 2] which proof that fusion of weak learners outputs achieves higher accuracy than a single classifier. The success of such ensemble methods relies on diversity, a concept which tries to explain how the differences between classifiers of an ensemble can improve the accuracy of the global predictor. This paper presents a genetic method to design classifier ensembles as an optimization process, considering the diversity [11] and the accuracy of the ensemble at each stage of its creation. The next section of this paper presents several references from the literature about diversity in classifier ensembles and describe several implementations using explicit formulation of diversity to train ensemble of classifiers. The third section presents the GenDiv algorithm, a GA-based implementation to design a classifier ensemble as an optimization process. Evaluation of the proposed implementation on several UCI Machine Learning datasets and comparisons with the accuracy of a single classifier are given in the forth section. The last section of this paper is dedicated to conclusions and possible improvements of the proposed algorithm.

L. Magdalena, M. Ojeda-Aciego, J.L. Verdegay (eds): Proceedings of IPMU'08, pp. 798–805
Torremolinos (Málaga), June 22–27, 2008

## 2 Diversity in classifier ensembles

The question one can ask is *how a set of weak learners whose individual accuracy is slightly superior to a random guess can perform better than the single best classifier when combined into an ensemble?* The answer to this question can be partially explained using the concept of diversity. If all the classifiers in the ensemble were identical, the accuracy of the whole ensemble would be equal to the one of its individual members. To perform better than a single classifier, the samples which are misclassified must be different from one individual learner to another. The algorithm that best illustrates this principle is probably the Adaboost algorithm [7]. Adaboost builds an ensemble by training individual classifiers in a cascading style, so that the current classifier focus its training on the samples that were misclassified by the previous classifiers of the ensemble. Adaboost has proven to be one of the most efficient ensemble algorithm. It enforces diversity by focusing on the complementarity between the individual classifiers of the ensemble. Several diversity metrics can be found in the literature. In [11] the authors expose the most widely used diversity measures. In this paper, diversity is quantified using the disagreement measure exposed in [9]. Table 1 gives the incidence matrix build on oracle outputs of two classifiers $C_i$ and $C_j$.

Table 1: Incidence matrix of oracle outputs for two classifiers.

| | $C_j$ correct(1) | $C_j$ wrong(0) |
|---|---|---|
| $C_i$ correct(1) | $a$ | $b$ |
| $C_i$ wrong(0) | $c$ | $d$ |

The disagreement measure between $C_i$ and $C_j$ is equal to the probability that the two classifiers disagree on their decisions:

$$D_{i,j} = b + c \tag{1}$$

Values of $D_{i,j}$ close to 1 signify higher diversity. Another frequently used measure is the Interrater Agreement $\kappa$. For $c$ class labels, $\kappa$ is defined on the $c \times c$ coincidence matrix $M$ of the two classifiers. The entry $m_{k,s}$ of $M$ is the probability that $C_i$ labels an object as $\omega_k$ when $C_j$ labels it as $\omega_s$. The agreement between $C_i$ and $C_j$ is given by:

$$\kappa_{i,j} = \frac{\sum_k m_{kk} - ABC}{1 - ABC} \tag{2}$$

where $\sum_k m_{kk}$ is the observed agreement between the classifiers and ABC is the *agreement-by-chance*

$$ABC = \sum_k \left( \sum_s m_{k,s} \right) \left( \sum_s m_{s,k} \right) \tag{3}$$

Low values of $\kappa$ signify higher disagreement and hence higher diversity. $D_{i,j}$ and $\kappa_{i,j}$ are pairwise measures between two classifiers. To generalize the diversity to the whole ensemble, the measure is averaged across all pairs of classifiers. In this paper, the diversity $Div_e$ of the whole ensemble is expressed as:

$$Div_e = \frac{2}{L\,(L-1)} \sum_{i=1}^{L-1} \sum_{j=i+1}^{L} D_{i,j} \tag{4}$$

where $C_i$ and $C_j$ are members of ensemble $e$.

In [13], the authors propose to use Kappa-Error Diagram, a two dimensional scatter plot of pairwise accuracy/diversity values, to study the diversity of different ensemble algorithms. Several implementations to explicitly use diversity while training a classifier ensemble have been proposed. In [4], the authors define a topology of algorithms and methods related to diversity in classifier ensembles and how to achieve it. In order to create diversity between classifiers, both the structure of the learner and the training data can be modified. However, it has been noticed that reshaping the data generally performs better than manipulations on architecture [16]. Acting on the training data is only effective for unstable classifiers, like decision trees or neural networks, for which small changes in the training data can lead to significant differences in the learned target function. Most of the diversity methods exposed in the literature use these types of base learner, contrary to nearest neighbour classifiers for instance, for which the learned decision boundaries are

robust to data alteration over the training set. Ensemble algorithms are used together with a combiner for supervised classification, to perform fusion of the label outputs of the different members of the ensemble. The combiner is generally a majority voting, weighted or not, but an additional classifier can be used to obtain the output of the whole ensemble. Various combination methods are reported in [10].

Apart from the method used to diversify the learners in the ensemble, a distinction can be established between ensemble algorithms that make an implicit or explicit use of diversity while training the ensemble. Bagging [2] for instance, belongs to the implicit methods. A certain number of base learners are trained with randomly chosen samples, but the diversity of the ensemble is observed *a posteriori*. Althought the Adaboost algorithm does not manipulate a diversity metric during the design stage of the ensemble, it belongs to the explicit methods, because samples distribution is no longer chosen randomly, but with the objective of maximizing the complementarity between the elements of the ensemble, by focusing on previously misclassified samples. Another explicit ensemble method is the DECORATE algorithm described in [14]. DECORATE iteratively trains ensemble members by adding artificial samples to the original training data. Those artificial samples are generated according to a metric including diversity in order to maximize the diversity of the whole ensemble. Bagging and boosting are meta-algorithms and can be used with any type of base learner. The Random Forest algorithm has been designed for decision trees. Random Forest [3] create ensembles of decision trees trained on a randomly chosen subset of features. In [12], the authors propose the Rotation Forest algorithm, and perform Principal Component Analysis (PCA) on the feature subsets to enforce diversity. They also present a detailed comparison between Bagging, Boosting, Random Forest and their implementation. The authors conclude that diversity is not the only important factor when designing classifier ensembles, and show that the performances of the ensemble strongly rely on the individual accuracy of its members. The next section of this paper presents the GenDiv algorithm. GenDiv is a GA-based implementation designed to optimize both individual accuracy and diversity of ensemble members.

## 3 The GenDiv Algorithm

Genetic algorithms [8] are optimization heuristics inspired by evolutionary biology and used when there is no exact method with reasonnable complexity for a given problem. Starting from a set of initial individuals, the population is optimized using reproduction, crossover and mutation operations, until a maximum number of iteration has been reached, or when there is no significant improvement over a certain number of iterations. Genetic algorithms require a coding scheme and a fitness function. The coding scheme, generally a binary string, is used to obtain a genetic representation of the solution domain, and the fitness function is used to evaluate the solution domain and select the individuals that will breed the next generation of solutions.

The GenDiv algorithm aims at designing a classifier ensemble by selecting from a pool of classifiers the subset of elements that best optimize a fitness function. Prior to the optimization process, the GenDiv algorithm builds a pool of base learners by training $N$ classifiers on a random redistribution of the original training set. By sampling examples from the original training set with replacement (also called bootstrap sampling), it is likely that the individual classifiers within the pool will exhibit some diversity. Algorithm 1 gives the pseudo code used to build the pool of base classifiers, and the pseudo code used to perform the evolutionary step is described in algorithme 2. Before the first iteration, the $N$ classifiers from the pool built previously are randomly grouped into $P$ ensembles of $L$ classifiers. Each ensemble is then given a fixed-length array of $L$ integers as genome. Those integer genes represent the indexes of the base

**Algorithm 1** Initialisation step of the GenDiv algorithm

**Require:**

$X^{tr}$ : the training samples (an $n \times p$ matrix)
$X^{te}$ : the test samples (an $m \times p$ matrix)
$N$ : the number of classifiers in the resulting pool

**Ensure:**

$Pool$ : a pool of $N$ base classifiers
$Outputs$: an $m \times N$ matrix containing the labels computed by each classifier on $X^{te}$

$Pool \leftarrow \emptyset$
**for** $i = 1$ to $N$ **do**
- draw a bootstrap sample $X_i^{tr}$ from the original training set $X^{tr}$
- build base classifier $C_i$ using $X_i^{tr}$ as training set
- $Pool \leftarrow Pool \cup C_i$
- retrieve the $m \times 1$ vector $\hat{Y}_i$ containing the class labels computed by $C_i$ on $X^{te}$
- store $\hat{Y}_i$ in the output matrix: $Outputs(:, i) \leftarrow \hat{Y}_i$

**end for**
**return** $Pool$ along with $Ouputs$

classifiers from the pool that are used in each ensemble, so that the evolutionary algorithm can breed the optimal classifier ensemble by selection and inheritance of the genes that best satisfy the fitness function. The fitness function used by the GenDiv algorithm and computed for all candidate ensembles at each iteration is given by:

$$Fitness_e = \alpha \cdot Acc_e + (1 - \alpha) \cdot Div_e \qquad (5)$$

where $Acc_e$ is the accuracy of ensemble $e$ using majority voting as combiner, and $Div_e$ is the diversity of ensemble $e$, given in Eq. (4). The more important the fitness of a given candidate ensemble is, the more often this ensemble will be selected for reproduction. Each pair of ensembles selected for reproduction produces two successors using a single-point crossover operation described in figure 1. The genome of each parent, that is, the list of base classifiers that form the ensemble, is randomly split in two parts, so that each successor inherits a part of each of its parents genome. To avoid multiple appearances of a given classifier in the same ensemble, duplicated genes are randomly replaced by classifiers from the initial pool, but that do not already exist in the genome of the ensemble. This feature, along with random mutations, can also help the evolutionary algorithm to escape local minina.

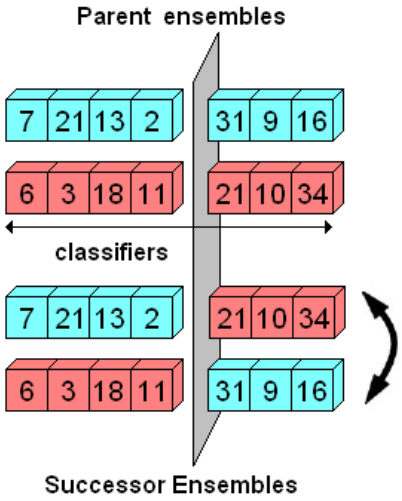


Figure 1: Illustration of the crossover

At each iteration, GenDiv replaces the population of $P$ ensembles by their $P$ successors (each pair of individuals breeding exactly two new ensembles), so that the population size remains constant. To avoid a general deterioration of the population, the new ensemble which possesses the smallest fitness is systematically replaced by the ensemble having the highest fitness in the previous breed. This evolution step is repeated until a maximum number of generations is reached or until no more improvement occurs dur-

**Algorithm 2** Evolutionary step of the GenDiv algorithm

**Require:**

$P$ : the number of ensembles to be optimized
$L$ : the number of classifiers in each ensemble
$Pool$ : the pool of $N = L \times P$ classifiers provided during initialisation step
$Outputs$ : the output matrix computed during initialisation step
$Y$ : an $m \times 1$ vector containing the true class label for each test sample
$MaxGen$ : the number of generations used for genetic optimization
$mr$ : the mutation rate of the genetic algorithm
$\alpha$ : the coefficient used in the fitness function given in equation 5

**Ensure:**

$E^*$: the optimal ensemble resulting from the last generation

**// Random selection of the initial population**
**for** $i = 1$ to $P$ **do**
- randomly select $L$ distinct classifiers from $Pool$ without replacement
- denote by $Genome_i$ the base learners selected from $Pool$ to build ensemble $E_i$
- compute $Acc_i$ by Majority Voting from $Outputs$ and $Y$
- compute $Div_i$ as defined in equation 5
- $Fitness_i \leftarrow \alpha \cdot Acc_i + (1 - \alpha) \cdot Div_i$

**end for**
**// Selection, reproduction and mutation of successive breeds**
$iter \leftarrow 1$
**while** $(iter \leq MaxGen)$ **do**
- denote by $E^+$ the ensemble which possesses the highest fitness value
- let $Parents$ be a list of $P$ ensembles selected (possibly multiple times) according to their fitness
- arrange $Parents$ to form pairs of distinct ensembles $E_iE_j$
- **for all** (pairs $E_iE_j$ of ensembles in $Parents$) **do**
  - build two new ensembles $E'_{ij}$ and $E'_{ji}$ using crossover and mutation operators
  - update accuracy, diversity and fitness values of $E'_{ij}$ and $E'_{ji}$
- **end for**
- denote by $E^-$ the ensemble with the lowest fitness value in the new population
- $E^- \leftarrow E^+$
- $iter \leftarrow iter + 1$

**end while**
**return** $E^*$ the optimal classifier ensemble in the last generation

ing several successive iterations. When evolution is complete, the algorithm returns the ensemble which possesses the highest fitness in the remaining population. The detailed pseudo code of the genetic implementation is given in algorithm 2. GenDiv requires heavy computation during its initialization, because $P \times L$ base classifiers must be trained and their respective outputs must be computed on additional tests samples. Since the evolution procedure consists in switching classifiers between ensembles until the best solution is found, no more training or output computation is required during this stage. So, the evolution of initial candidate solutions completes faster than initialization. In the next section of this paper, we present results obtained with the GenDiv Algorithm on several UCI datasets[1] and discuss the influence of diversity and individual accuracy when designing classifier ensembles.

## 4 Experimental material

UCI datasets used during experiments are reported in Table 2. The base classifiers used for experiments are decision trees, because they are sensitive to redistributions of the original training set, and thus exhibit diverse errors. The GenDiv algorithm is implemented using Matlab 6.5, which also provide a built-in toolbox to design decision trees.

Table 2: Characteristics of the UCI datasets.

| Dataset | Classes | Samples | Features |
|---|---|---|---|
| Credit | 2 | 690 | 15 |
| Pendigits | 10 | 10992 | 16 |
| Waveform | 3 | 5000 | 40 |

The GenDiv algorithm is run on 5 ten-fold cross validations to breed a population of $P = 10$ classifier ensembles, each consisting of $L = 10$ decisions trees taken from an initial pool of 100 classifiers. Results reported in this section are averaged across the 50 testing accuracies obtained over the different runs. When the datasets from the UCI consist in separate training and test files, training and test samples are used altogether for cross validation, so that 90 percent of the available data are always used to train the base classifiers. The remaining 10 percent are used to compute the label outputs of each base learner on additional test samples and to perform the evolutionary step of the GenDiv algorithm. For all UCI datasets, the mutation rate and the coefficient used in the fitness function given in Eq.(5) are set respectively to $mr = 0.001$ and $\alpha = 0.75$. The value given to $\alpha$ ensures that a more important role will be dedicated to accuracy rather than diveristy during the desing stage of the ensemble, since previous studies have already shown that optimizing diversity alone generally lowers the accuracy of classifier ensembles. For the Credit dataset, which possesses a small number of samples, the size of the bootstrap sample is the same as the size of the initial training set. For pendigits and waveform datasets, the size of the bootstrap samples used for training each member of the ensembles is reduced to 2000 and 750 samples respectively, to alleviate the computation required during initialization while providing a sufficient number of samples to accurately train each base learner. A large number of samples is required for the genetic optimization to perform effectively. The initial bagging step may not always generate enough diversity, and in this case, the GA-based optimization will perform poorly. We set the maximum number of generation to 20 and plot the evolution of ensemble solutions for the pendigits dataset in figure 2.

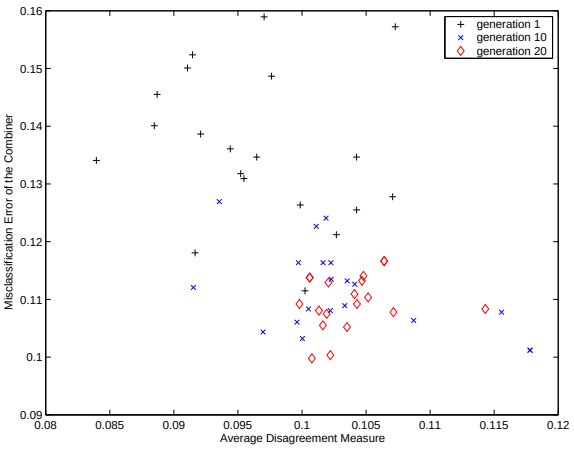


Figure 2: Evolution of candidate ensembles for the pendigits dataset

Figure 2 draws the evolution of a set of classifier ensembles during the GA-based optimization. Each point represent a classifier ensemble, defined by its misclassification error and diversity. It shows that the GenDiv algorithm progressively lowers the misclassification error of initial candidate ensembles while diversity is optimized. In this example, the GenDiv algorithm can be seen as a good optimization step when used together with bootstrap resampling. Table 3 compares the classification error of the GenDiv algorithm after 100 generations with the performances obtained with Bagging and a single classifier (SB) of the same nature. In this study, a single classifier is a unique decision tree trained on the whole training set. The Bagging algorithm can be seen as a random selection of $L = 10$ decision trees from the initial pool of classifiers. Results show that GenDiv can efficiently optimize classifier ensembles in terms of misclassification error. This study also confirms that diversity and individual accuracy of ensemble members must both be considered when designing classifier ensembles.

Table 3: Classification Error for GenDiv, Bagging and Single Best classifier.

| **Dataset** | **GenDiv** | **Bagging** | **SB** |
|---|---|---|---|
| Credit | **12.97±1.55** | 14.13±1.30 | 17.25±2.02 |
| Pendigits | **10.30±0.74** | 11.24±0.12 | 13.03±1.30 |
| Waveform | **14.40±0.94** | 17.68±1.09 | 15.22±0.40 |

## 5 Conclusion and future works

This paper presents the GenDiv algorithm, a genetic implementation which considers both accuracy and diversity during the design stage of classifier ensembles. Bootstrap resampling is used to generate a set of candidate ensembles which are optimized using a genetic heuristic. This method minimizes the misclassification error of the ensemble while maintaining diversity between its members. Results obtained with several UCI benchmarks show that it is superior or at least competitive to bagging and that the designed ensembles perform better than a single classifier of the same nature. Experiments also confirm that indivual accuracy plays an important role when designing classifier ensembles, and that diversity is not the only factor that must be considered. The main drawback of the GenDiv implementation is the large number of random classifiers that must be generated during the initialization step to provide a sufficient number of candidate ensembles. The GA-based optimization also rely on the initial diversity generated with bootstrap resampling. For this reason, the GenDiv algorithm is more likely to perform efficiently for large datasets. The disagreement value used in the fitness function is a pairwise measure, averaged across all pairs of classifiers to express the diversity of the whole ensemble, which may reveal not as accurate as using a non pearwise metric applied on the whole ensemble, or as adding classifier one by one to the ensemble focusing on diversity, the way Adaboost operates. Futur improvements will include a replacement of the diversity measure used in the current GenDiv implementation, as well as a modification of the algorithm to increase the quantity of initial diversity using extraction of feature subsets. Further investigations about using diversity during the design stage of classifier ensembles are also planned.

### Acknowledgements

The present research work has been supported by the European Community, the Délégation Régionale à la Recherche et à la Technologie, the Ministère de l'Education Nationale, de la Recherche et de la Technologie, the Région Nord-Pas de Calais, the Centre National de la Recherche Scientifique. The authors gratefully acknowledge the support of these institutions.

## References

[1] A. Asuncion and D. J. Newman (2007). *UCI Machine Learning Repository.* http://www.ics.uci.edu/ mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

[2] L. Breiman (1996). Bagging Predictors. In*Machine Learning*, volume 24, no. 2, pages 123-140, 1996.

[3] L. Breiman (2001). Random Forests. In *Machine Learning*, volume 45, no. 1, pages 5-32, 2001.

[4] G. Brown, J. Wyatt, R. Harris and X. Yao. (2005). Diversity creation methods: a survey and categorisation. In *Information Fusion*, volume 6, no. 1, pages 5-20, 2005.

[5] R. O. Duda, P. E. Hart and D. G. Stork (2001). *Pattern Classification (2nd edition)*. Wiley, New York.

[6] Y. Freund, R.E. Schapire (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML-96)*, pages 148-156, July 1996.

[7] Y. Freund and R.E. Schapire (1997). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In *J. Computer and System Sciences*, volume 55, no. 1, pages 119-139, 1997.

[8] D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA.

[9] T. K. Ho (1998). The random space mathod for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 20, no. 8, pages 832-844, 1998.

[10] L. I. Kuncheva (2004). *Combining Pattern Classifiers*. John Wiley & Sons.

[11] L. I. Kuncheva and C. Whitaker (2003). Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. In *Machine Learning*, volume 51, no.2, pages 181207, 2003.

[12] J. J. Rodriguez, L. I. Kuncheva and C. J. Alonso (2006). Rotation Forest: A New Classifier Ensemble Method. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 28, no. 10, pages 1619 - 1630, October 2006.

[13] D. D. Margineantu and T. G. Dietterich (1997). Pruning Adaptive Boosting. In *Proceedings of the 14th International Conference on Machine Learning* pages 211-218, 1997.

[14] P. Melville and R.J. Mooney (2005). Creating diversity in ensembles using artificial data. In *Information Fusion*, volume 6 no. 1, pages 99-111, 2005.

[15] T. Mitchell (1997). *Machine Learning*. McGraw Hill.

[16] D. Partridge and W.B. Yates (1996). Engineering multiversion neural-net systems. In *Neural Computation*, volume 8, no. 4, pages 869-893, 1996.