

# Ensemble Learning Using Multi-objective Evolutionary Algorithms

ARJUN CHANDRA and XIN YAO

*The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, United Kingdom*  
(a.chandra|x.yao@cs.bham.ac.uk)

May 18, 2005

**Abstract.** Multi-objective evolutionary algorithms for the construction of neural ensembles is a relatively new area of research. We recently proposed an ensemble learning algorithm called DIVACE which is an acronym for DIVERse and ACCurate Ensemble learning algorithm. It was shown that DIVACE tries to find an optimum trade-off between diversity and accuracy as it searches for an ensemble for some particular pattern recognition task by treating these two objectives explicitly as multi-evolutionary pressures. A detailed discussion of DIVACE together with further experimental studies form the essence of this paper. A new diversity measure which we call Pairwise Failure Crediting (PFC) is also proposed here. This measure forms one of the two evolutionary pressures being exerted explicitly in DIVACE. Experiments with this diversity measure as well as comparisons with previously studied approaches are hence considered. Detailed analysis of the results show that DIVACE, as a concept, has promise.

**Keywords:** ensemble learning, diversity, neural networks, multi-objective, neuroevolution

## 1. Introduction

Ensembles of learning machines have been formally and empirically shown to outperform (generalise better than) single predictors. Tumer and Ghosh [46] present the formal proof of this. According to Dietterich [26], ensembles form one of the main research directions as far as machine learning research is concerned. Brown [12] gives an extensive survey of ensemble methods. A theoretical account of why ensembles perform better than single learners is also presented here [12]. Dietterich [26] also elucidates some reasons for performance enhancements/better generalisation using ensembles by exploiting the notion of machine learning algorithms as being search algorithms which seek the most accurate hypothesis (hypothesis in this case being the learner/predictor which learns an input-output mapping function) in a space of possible hypotheses.

Although ensembles perform better than their members, constructing them is not an easy task. As Dietterich [26] points out, the key



© 2005 Kluwer Academic Publishers. Printed in the Netherlands.

to successful ensemble methods is to construct base models (individual predictors) which perform better than random guessing individually and which are at least somewhat uncorrelated as far as making errors on the training set is concerned. The statement essentially tells that in order for an ensemble to work properly it should have a diverse and accurate set of predictors (also mentioned in one of the seminal works on diversity in classifier ensembles by Hansen and Salamon [28]). Krogh and Vedelsby [33] formally show that an ideal ensemble is one which consists of highly correct (accurate) predictors but at the same time disagree as much as possible (i.e. substantial diversity amongst members is exhibited). This has also been tested and empirically verified as referred to in [39] and shown in [40]. Thus, diversity and accuracy are two key issues that should be taken care of when constructing ensembles. There exists a trade-off as to what should be the optimal measures of diversity and accuracy [16].

Also, as stated in [16], incorporating evolution in segments of machine learning has been a widely studied area (e.g. evolving neural networks [10, 44, 47]) with the main thrust here being that evolution has been used as another fundamental form of adaptation in addition to learning, making neural systems adapt to a dynamic environment effectively and efficiently [47]. This has its roots in the ‘baldwin effect’ [8]. Not surprisingly, evolution has also been used in order to construct ensembles [3, 30, 34, 36, 40].

Abbass [7] proposed the Pareto-frontier Differential Evolution (PDE) method, which is an extension of the Differential Evolution (DE) algorithm proposed by Storn and Price [45]. In [1], an algorithm for ensemble learning called Memetic Pareto Artificial Neural Network (MPANN), which is a customised version of PDE for evolving neural networks, was proposed.

Recently we proposed an algorithm called DIVACE (DIVERse and Accurate Ensemble Learning Algorithm) [16, 15] which uses good ideas from Negative Correlation Learning (NCL)[35] and Memetic Pareto Artificial Neural Network (MPANN)[1, 3], and formulates the ensemble learning problem as a multi-objective problem explicitly within an evolutionary setup, aiming at finding a good trade-off between diversity and accuracy. An explicit treatment of both diversity and accuracy within an evolutionary setup for constructing ensembles was also carried out by Opitz and Shavlik [40], however, these two objectives were merged into one here.

One very strong motivation for the use of evolutionary multi-criterion optimisation in the creation of an ensemble in both DIVACE[16] and MPANN [3] is that due to the presence of multiple conflicting objectives, the evolutionary approach engenders a *set* of near optimal

solutions. The presence of more than one optimal solution indicates that if one uses multi-objectivity while creating ensembles, one can actually generate an ensemble automatically where the member networks would inadvertently be near optimal [16].

This paper presents DIVACE with all its intricacies and establishes ensemble learning as a multi-objective evolutionary learning process that aims to produce a set of near optimal learners which, when combined, lead to the formation of a competitive aggregate learner. We further present some experimental results and try to show that DIVACE is a conceptual tool which can be used to construct ensembles regardless of the manner in which the objective function evaluation is carried out. As long as the basic framework of the algorithm and the fact that function evaluation involves the explicit evaluation of accuracy and diversity for the individual members are taken care of, DIVACE would lead to a competitive ensemble. We show this by introducing a new diversity measure which we call Pairwise Failure Crediting (PFC) and use this within the DIVACE framework. In the following section, we consider why multi-objective evolutionary algorithms can be of interest for ensemble researchers. We then briefly describe the notions of diversity and accuracy and the trade-off they exhibit with regard to ensembles. This is followed by a detailed discussion of DIVACE and its intricacies. We then propose the PFC diversity measure. Results from DIVACE and a brief discussion thereof are further considered.

## 2. Multi-objective Evolutionary Algorithms in Ensemble Design

Multi-objectivity in neural network ensembles, as an area of research, has not been explored extensively yet. The idea of designing neural networks within a multi-objective setup was first considered by Abbass in [4]. Here, the multi-objective problem formulation essentially involved setting up of two objectives viz. complexity of the network and the training error (quadratic error/mean square error). The network complexity here could mean the number of synaptic weights, number of hidden units or a combination of both. An algorithm called MPANN was proposed here which uses Pareto differential evolution [7] which is supposed to have shown improvements on many other multi-objective evolutionary algorithms. MPANN was later on considered in [3, 5] for learning and formation of neural network ensembles, albeit, with a different multi-objective formulation (as opposed to that in [4]).

The main reason for using a multi-objective evolutionary approach to designing ensembles is that multi-objectivity enforces the search/

optimisation process (the search process here being finding neural networks with good overall performance) to yield a set of near optimal solutions instead of just a single solution. Getting a set of solutions essentially means that we get a set of near optimal neural networks. These near optimal neural networks could in turn be used as members of an ensemble. Additionally, if we use evolutionary multi criterion optimisation for designing ensembles, we would, in fact, be speeding up the search process for the very reason that evolutionary approaches are population based approaches. With a population based approach we will, in essence, be generating a set of networks and the underlying multi-objective framework would take care of the selection of a set of near optimal solutions/networks from the population. The whole process of generating a pareto set of neural networks which could be used as members of an ensemble will be automatic as the whole population (which includes the pareto set of individuals/networks) would, with the passage of time, move towards the pareto front. Thus, the idea of using multi-objective evolutionary algorithms for the purpose of designing neural network ensembles seems very promising.

The main problem in actually using such an approach is the formulation of the multi-objective optimisation problem such that not only the final ensemble created have accurate members but the members also be uniformly distributed on the pareto optimal front i.e. diversity is catered for (as discussed in the next section). Remarkably, for multi-objective optimisation to be effective, the optimisation process should lead to convergence to the pareto optimal front while at the same time maintaining as diverse a distribution of the near optimal solutions as possible on the pareto front [24]. A striking parallel between multi criterion optimisation and the necessity of having diverse enough members within an ensemble is evident. Hence, formulating a problem properly would surely do justice to both (multi-criterion optimisation and ensembles as disparate yet related computational paradigms).

### 3. Diversity and Accuracy in Ensembles

#### 3.1. DIVERSITY

Brown et al. ([12]) give a good account of why diversity is necessary in neural network ensembles and present a taxonomy of methods that enforce it and which are in practise. If two predictors make different errors on the same data points/inputs, they are said to be diverse. In essence, ensemble members which initially begin with different weight initialisations, after training over a particular dataset, are likely to give

different errors for any particular data point. The outputs are then combined, say for instance in the simple averaging case, which leads to the mean output being closer to the expected value. This is what is done by various diversity promoting mechanisms.

### 3.2. ACCURACY

Accuracy can be defined as the property of a network (ensemble member) performing better than random guessing on a new input (Brown et al. [12]). In the regression context, the lower the mean square error of a network member on any given input, the more accurate it is. Similarly, in the classification context, if we take the output of individual networks as the posterior probabilities of classes, we can apply the quadratic error function here as well, which implies that the lower this error for any given input, the more accurately a network classifies it.

### 3.3. THE TRADE-OFF

Many ensemble methods have their formal grounding in Krogh and Vedelsby's *ambiguity decomposition* [33]. This has been shown in [11, 14, 13]. It expresses the relationship between the mean square error of the ensemble and the average of the mean square errors of its members. More precisely, it states that the quadratic error of the ensemble is guaranteed to be less than or equal to the average of the quadratic errors of the individual members. This can mathematically be given as,

$$(f_{ens} - \Phi)^2 = \sum_i w_i (f_i - \Phi)^2 - \sum_i w_i (f_i - f_{ens})^2 \quad (1)$$

and,

$$(f_{ens} - \Phi)^2 \leq \sum_i w_i (f_i - \Phi)^2 \quad (2)$$

In the above equations,  $f_{ens}$  is the ensemble output,  $\Phi$  is the desired or target output,  $f_i$  is the output of the  $i^{th}$  member in the ensemble,  $w_i$  signifies how much an individual member contributes to the ensemble (with the constraint  $\sum_i w_i = 1$ ), and, the ensemble output is defined as  $f_{ens} = \sum_i w_i f_i$ .

In Equation 1, the second term on the right hand side (i.e.  $\sum_i w_i (f_i - f_{ens})^2$ ) will always be greater than or equal to zero i.e. positive or zero which makes the left hand side of the equation less than or equal to the right hand side (as shown in Equation 2). This is often called as the ambiguity term and it tells how different the individual members within

the ensemble are for some test pattern (data point) i.e. emphasises diversity. It means that if this term is somehow made large, the left hand side will become very small i.e. the error estimate of the ensemble for a given data point will decrease. In other words, if we have more diversity in the ensemble, the accuracy of the combined predictor will increase. It can also be seen that if we try to maximise this term i.e. cross some limit (lets call it a trade-off line), the first term (i.e.  $\sum_i w_i (f_i - \Phi)^2$ ) would also increase, which then would override the whole effect caused due to the amplification of the ambiguity term. Here, the first term signifies the accuracy of an individual for some test pattern. **This signifies that we should have diverse members in the ensemble but there is a limit/trade-off line crossing which would make the individual predictors less accurate.** This is what is often referred to as the **trade-off between diversity and accuracy** in ensembles. In the classification context, according to Hansen and Salamon [28] and as also mentioned in [12], a necessary and sufficient condition for a majority voting ensemble of classifiers to be more accurate than any of its component classifiers is that the components be both accurate and diverse. So the trade-off applies to both regression and classification problems.

Brown and Wyatt [14] show that there is a family of methods which essentially exploits the ambiguity decomposition, with each method subtly using it in some aspect of ensemble construction. The relationship between NCL [35] and ambiguity decomposition has been scrutinised in [11, 14, 13] where it was shown that the error function used in NCL [35] can be directly modelled or transformed into the ambiguity decomposition formulation if some assumptions are taken into account. Not only NCL [35], Krogh and Vedelsby [33] came up with their own ensemble learning scheme directly utilising ambiguity in the training process as well as in fusing the decisions of the ensemble members. Moreover, feature selection using the ambiguity term of the decomposition has also been researched on by Opitz [38] where a genetic algorithm was used to select features with an ambiguity based fitness function.

The ADDEMUP algorithm by Opitz and Shavlik [40] is another example where ambiguity has been explicitly used. Here, a GA was used to evolve a neural network population from which an ensemble is formed in every generation by pruning the population to a select few or  $N$  fittest members. This pruning essentially utilises the ambiguity decomposition in that, the ambiguity term explicitly forms an additive part of the fitness function (the other part being accuracy). This function also uses a parameter which balances accuracy and ambiguity (i.e. scales the ambiguity term) and has been viewed as managing the

trade-off between these two entities. Ambiguity can be referred to as diversity as well. Incidentally, as shown in [14], NCL [35] also uses a trade-off parameter in the error function which tries to find the right balance between accuracy and diversity i.e. weighing the emphasis on diversity. Moreover, Opitz [38] also uses the balancing parameter to alter the emphasis on diversity.

Additionally, Brown et al. [12] and Chandra and Yao [16] have also described the relationship between accuracy and diversity from the individual member output distribution perspective. It is believed that the more diverse the members are, the better the outputs of each will be distributed around the desired output. Also, the more accurate the members are, the closer will their outputs be to the desired output. Hence, describing the trade-off qualitatively, having compactly well distributed outputs [16] around the desired output would result in a better expectation of the desired value when the decisions from the members are fused.

#### 4. Diverse and Accurate Ensemble Learning Algorithm (DIVACE)

##### 4.1. ORIGINAL PROBLEM FORMULATION

Given a data set on which to make the ensemble work, the following formulations are considered as the two objectives needed to be taken care of in order that the final ensemble generalises well:

**Accuracy.** Given a training set  $T$  with  $N$  patterns. For each network  $k$  in the ensemble,

$$\text{(Minimise) Accuracy}_k = \frac{1}{N} \sum_{i=1}^N (f_k^i - o^i)^2, \quad (3)$$

where  $o^i$  is the desired output and  $f_k^i$  the posterior probability of the class (classification task) or the observed output (regression task) for one training sample  $i$ .

**Diversity.** From NCL, the correlation penalty function is used as the second objective on which to optimise the ensemble performance. Let  $N$  be the number of training patterns and let there be  $M$  members in the ensemble, so for each member  $k$ , the following term gives an

indication of how different it is from other members.

$$\text{(Minimise) Diversity}_k = \sum_{i=1}^N (f_k^i - f^i) \left[ \sum_{j \neq k, j=1}^M (f_j^i - f^i) \right], \quad (4)$$

where  $f^i$  is the ensemble output for a training sample  $i$ .

#### 4.2. THE ALGORITHM: DIVACE

Following is the DIVACE algorithm:

**Step 1:** Create a random initial population<sup>1</sup> (size  $M$ ) of networks, the weights for each assigned uniformly distributed random values  $U(0, 1)$ .

**Step 2:** Apply Back-Propagation (BP) to all individuals in the population.

**Step 3:** Repeat until termination condition (a certain number of generations in our case)

1. Evaluate the individuals in accordance with the two objective functions and label the non-dominated set (Non-dominated sorting algorithm used here.)
2. If the number of non-dominated individuals is less than 3 then a repair rule similar to that used in MPANN (Abbass [5]) is used.
3. All dominated solutions are deleted from the population.
4. Repeat until population size is  $M$ 
  - Variance update: updating the variance value for the Gaussian distribution used in crossover. We do it according to,

$$\sigma^2 = 2 - \left( \frac{1}{1 + e^{(\text{anneal\_time} - \text{generation})}} \right) \quad (5)$$

where `anneal_time` is a parameter signifying exploration time/ number of generations for which the search process is to be explorative after which the value of  $\sigma^2$  decreases exponentially to finally reach a fixed value of 1 and it remains 1 until the final iteration. In our experiments, we use a value of 50 for the `anneal_time` parameter.

---

<sup>1</sup> For training, we take all the networks in the population as our ensemble but for testing, we only use the final pareto set as the ensemble.

- Select 3 parents at random from the population. Let  $\alpha_1$  be the main parent and  $\alpha_2$  and  $\alpha_3$  be the supporting parents.
- Perform crossover: Produce a child which has an architecture which is similar to the parents but weights given by,

$$w_{hi} = w_{hi}^{\alpha_1} + N\left(0, \sigma^2\right) \left(w_{hi}^{\alpha_2} - w_{hi}^{\alpha_3}\right) \quad (6)$$

$$w_{oh} = w_{oh}^{\alpha_1} + N\left(0, \sigma^2\right) \left(w_{oh}^{\alpha_2} - w_{oh}^{\alpha_3}\right) \quad (7)$$

- Perform mutation: Mutate the child with probability  $1/|pop|$  ( $|pop|$  being the size of the population) according to,

$$w_{hi} = w_{hi} + N(0, 0.1) \quad (8)$$

$$w_{oh} = w_{oh} + N(0, 0.1) \quad (9)$$

- Apply BP to child and add it to the population.

### 4.3. DIVACE INTRICACIES

Here we are going to discuss some important details about our algorithm such as the representation of neural networks (genotype-phenotype mapping), some aspects of training, the evolutionary process and finally some facts about the formation of the ensemble.

#### 4.3.1. Representation of the networks in the population

There have been many representation schemes developed in order to encode neural networks to facilitate their evolution. With DIVACE, since it uses PDE [1, 3, 4, 5, 7] and we have not really changed much of the evolutionary process, we stick to the trivial genotype-phenotype mapping in that, we consider our phenotype as our genotype. Thus our genotype is 2 weight matrices i.e. one matrix for synaptic weights between two layers of the network (our networks have 3 layers, hence 2 matrices per network). The main reason why there is no *special* genotypic representation is because in DIVACE we only evolve the weights and *not* the structure of the networks (the structure is fixed with 3 layers and 5 nodes for the hidden layer). The whole population of networks is initialised such that the weights for each network are uniformly distributed random values in  $U(0, 1)$ .

#### 4.3.2. Partial training and Lamarkian evolution

All the networks in the population are trained for 5 epochs using BP initially before moving on to the evolutionary stage of the algorithm.

During the evolutionary process, the new members generated as a result of the pareto differential evolution [7] technique are also trained for 5 epochs using BP before they are entered into the new generation. The idea here is that if we train the networks partially we essentially direct the search process towards the feasible region within the search space. It may somewhat look like a direct model of the *Baldwin effect* [8]. Actually, our approach i.e. the way we carry out the evolutionary process, is more like *Lamarckian evolution* which, in the natural sense, is a wrong theory of evolution as proved by Baldwin in [8] but has been shown to work with neural network learning in a multi-objective setting as is the case in [3, 4, 5].

Lamarckian evolution simply adds a feedback loop into the evolutionary process as far as the genetic code of a population is concerned. We train the networks so that the search process is directed towards the feasible region after which we recombine and mutate the networks in accordance with PDE and hence the new generation gets some *learned genetic information* (manifested in the weights of the parent networks). This simply takes care of the fact that new individuals/networks start at a point which is somewhere near the feasible region in the search space rather than starting at arbitrary points in this space. Feasible region here means a region close to the optimal region (region near the pareto optimal front) within the search space. The lamarkian feedback loop here is the weights of the parent networks which are directly used to create the offspring. This is more like learned genetic information propagation as opposed to passing on the unaltered genetic information (weights) which would not make any sense in this case.

As will be seen shortly, the way we generate the offspring is that we simply add some *noise* to the genotype of one of the parents in order to get the genotype of the child. This makes the generated child be in a position which is somewhat in the neighbourhood of the positions of the three (PDE uses three parents to create an offspring) parents used. Since the child is then trained using BP before putting it into the new generation (population) this learning further changes the genetic information which might later on be used to generate offspring for the next generation. Thus, the learning process here is very close to what happens in Lamarckian evolution.

#### 4.3.3. *Fitness evaluation*

This is done according to the non-dominated sorting procedure as given in [43]. Here we give a brief insight into the concept of non-domination followed by a brief account of the non-dominated sorting procedure.

Lets assume we have  $n$  objectives that need to be satisfied for the solution to a problem to be optimal. If we have two solutions from

the solution space of this problem, say  $sol_1$  and  $sol_2$ , we say that  $sol_1$  dominates  $sol_2$  iff:

1.  $sol_1$  is no worse than  $sol_2$  in all  $n$  objectives and,
2.  $sol_1$  is strictly better than  $sol_2$  in at least one objective

This implies that  $sol_1$  is nearer to the pareto optimal front as compared to  $sol_2$ ; it dominates  $sol_2$ . This is the concept of non-domination.

This sorting procedure is a part of the NSGA algorithm proposed by Srinivas and Deb [43]. It essentially classifies a set of solutions into groups or levels (called non-dominated fronts) where the lower the level, the more dominating a solution is.

Let there be  $S$  solutions in the solution space. If there are  $O$  objectives then each solution has  $O$  objective function values. The non-dominated sorting procedure can be enlisted as follows:

**Step 1:** Let  $i = 1$

**Step 2:** For all  $j = 1, 2, 3, \dots, N$  and  $j \neq i$ ,  $sol_i$  and  $sol_j$  are compared using the non domination conditions as follows:

---

$sol_i$  dominates  $sol_j$  iff,

1.  $sol_i$  is no worse than  $sol_j$  in all  $O$  objectives and,
2.  $sol_i$  is strictly better than  $sol_j$  in at least one objective

---

**Step 3:** If for any  $j$ ,  $sol_i$  is dominated by  $sol_j$ , we mark  $sol_i$  as *dominated*

**Step 4:** Increment  $i$ , i.e.  $i = i + 1$  and go to step 2 (until  $i > N$ )

**Step 5:** All solutions **not** marked *dominated* are the non-dominated solutions (i.e. are in the non-dominated front)

In our case there are two objectives: *accuracy* and *diversity*. After non-dominated sorting, we are left with a set of individuals/networks which are better than the rest of the networks in the population. This is the non-dominated set and while testing, we use this set as the ensemble. While training we consider the whole population as the ensemble mainly because the diversity objective calls for comparing each and every network in the population. It tries to make a network as different as possible from other networks in the population.

An important point to make here is that, it may sometimes happen that the non-dominated set has less than 3 individuals (the least number of individuals required to apply genetic operators) in which case

we simply run a *repair* algorithm until the number of individuals in the non-dominated set becomes greater than 3. The repair algorithm mainly looks into the next non-dominated front and includes individuals from this front in the set of non-dominated individuals. As a result, the final non-dominated set might have dominated individuals as well but in order to facilitate the evolutionary process, this step is necessary. However, other crossover operators can be used instead of the one used in PDE which would then not necessitate the presence of a lower limit on the number of individuals in the non-dominated set.

#### 4.3.4. *Evolutionary process*

The evolutionary process is quite similar to what is used in MPANN [1, 3, 5] and in PDE [7]. PDE was an extension of differential evolution [45] and is applicable in the case of multi criterion optimisation problems. In DIVACE, we have made this approach somewhat adaptive. We have tried to incorporate ideas from simulated annealing into the crossover operator which is one of the main genetic operators in any evolutionary process.

In DIVACE, after the fitness evaluation, we generate the mating pool. This mating pool mainly contains the individuals which are in the non-dominated set as determined by the fitness evaluation. The members which are not in the non-dominated set are removed from the population. Once we get the mating pool, we are ready to apply the genetic operators viz. selection, crossover and mutation and generate the new population of networks.

Following is a brief discussion of the genetic operators used in DIVACE.

**Selection.** In differential evolution, three parents are selected for both crossover and mutation. The same happens in PDE but the way in which these three are used changes. After the evaluation of the fitness which is done using the non-dominated sorting procedure, we get a non-dominated set of individuals/networks. As mentioned earlier, there have to be at least 3 individuals in this non-dominated set without which we would not be able to apply the genetic operators. In the selection process, we essentially select 3 individuals from the non-dominated set *randomly*. Random selection is employed mainly because any individual in one non-dominated front is no worse (and no better) than any other individual in this front. Although we do sometimes have to include individuals from the next non-dominated front into the mating pool, but this is done only when there are not enough members in the original non-dominated front, which then means that the mating pool would in essence have networks which are similar (or very nearly

similar) as far as their fitness is concerned. The selected parents are called  $\alpha_1$ ,  $\alpha_2$  and  $\alpha_3$  where  $\alpha_1$  is our main parent and  $\alpha_2$  and  $\alpha_3$  are our supporting parents.

**Crossover.** Moving on to the next genetic operator, we use an adaptive crossover method. In this, we incorporate the idea of simulated annealing into the process that generates the offspring. What we do is that we make the variance of the Gaussian distribution used to generate new individuals somewhat adaptive. In PDE, the use of a Gaussian distribution for crossover generates well-spread children around the main parent and along the directions of the supporting parents i.e. offspring generated are in the neighbourhood of the parents.

Since the parents are near the pareto front, the children will thus be somewhere near the pareto front as well (idea from Lamarkian evolution). The change we make in the crossover process is that we make it explorative to begin with. This means that the children will be more widely spread around the parents which generate them, yet these will be somewhere near the pareto region and might even help in the much needed diversity property as far as both the ensemble formation and multi criterion optimisation are concerned. This wider spread is achieved by making the variance i.e.  $\sigma^2$  vary with time (generations), eventually reducing to 1, as mentioned in Section 4.2.

Crossover is performed as follows:

$$w_{hi} = w_{hi}^{\alpha_1} + N(0, \sigma^2) (w_{hi}^{\alpha_2} - w_{hi}^{\alpha_3}) \quad (10)$$

$$w_{oh} = w_{oh}^{\alpha_1} + N(0, \sigma^2) (w_{oh}^{\alpha_2} - w_{oh}^{\alpha_3}) \quad (11)$$

where  $w_{hi}$  and  $w_{oh}$  are the weights (input to hidden layer and hidden to output layer respectively) of the child generated.

**Mutation.** Incorporating mutation into the offspring generation process is another difference between DIVACE and MPANN [1, 3, 5] and also PDE[7]. In fact, PDE was extended by Abbass in [2] where the mutation operator was used. This approach was called ‘self-adaptive pareto differential evolution’. With multi criterion optimisation and when using PDE, the offspring generated using crossover are in the neighbourhood of the parents. In order to facilitate having a good distribution (again to enforce diversity) mutation can play an important role. We mutate the child generated by the crossover process with a probability  $1/|population|$  ( $|population|$  is the size of the population) according to,

$$w_{hi} = w_{hi} + N(0, 0.1) \quad (12)$$

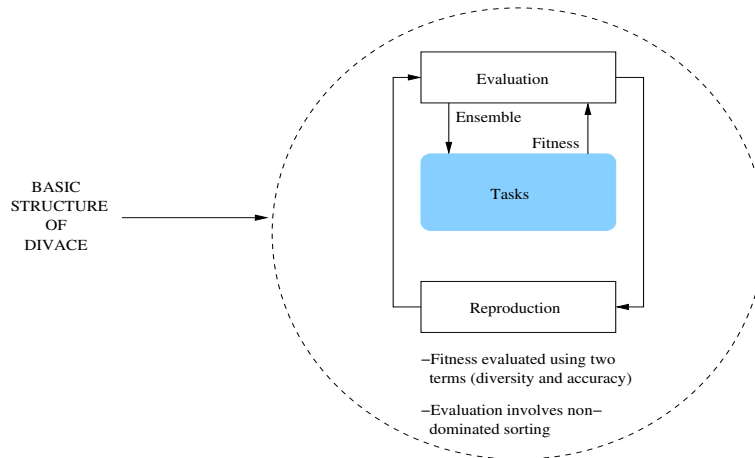


Figure 1. Basic structure of DIVACE.

$$w_{oh} = w_{oh} + N(0, 0.1) \quad (13)$$

#### 4.3.5. Ensemble formation

The final ensemble after the whole evolutionary process is the non-dominated set of individuals/networks in the final generation. This set is supposed to be very close to the pareto front and is automatically generated due to the multi-objective setting of DIVACE.

#### 4.4. DIVACE CONCEPT

As in any evolutionary algorithm, DIVACE has the usual phases of fitness evaluation and reproduction. The fitness evaluation is, however, not straight forward. It involves non-dominated sorting [43] of the population being evolved i.e. the members in the ensemble with the objectives being accuracy and diversity. Explicit enforcement of these two pressures helps achieve two goals. One is the fact that ensemble members need to be diverse which directly leads to the other goal of having a well spread distribution of solutions at the pareto front, a required property in multi-objective evolutionary algorithms. This explicit treatment of diversity together with accuracy is the main idea behind DIVACE. Figure 1 gives the basic structure of DIVACE.

## 5. New Diversity Measure: Pairwise Failure Crediting (PFC)

Although we have used NCL's penalty function term as the measure for diversity in our multi-objective formulation of the problem, we did mention in [16] that DIVACE is in no way limited to any particular term for the objective functions. As far as diversity measures are concerned, a lot of research has gone into defining what diversity really means and how one could, in fact, measure it. Most of the literature concerns population level diversity measures i.e. given a population of individuals, these measures can tell how diverse the population is. These could be used provided we are evolving the ensemble as a whole i.e. in every generation we replace the whole ensemble with a new one in which case the pareto set would contain many ensemble instances from which one can be chosen by an expert. However, this would not be as effective as evolving the members of an ensemble such that these members are a part of the pareto set which then would not necessitate the presence of an expert as the ensemble is automatically produced. This automatic construction makes measuring diversity at the individual level essential as, in the DIVACE sense, diversity is explicitly being treated as one of the objectives or evolutionary pressures. The dearth in the literature, as far as individual level diversity measures is concerned, led us to synthesise a new diversity measure which bears a strong resemblance to implicit fitness sharing [21, 27]. Although this measure can be used at both the individual as well as population/ensemble levels, here we are only concerned with the former.

We propose a new diversity measure here and use it within the DIVACE setup. We call this measure Pairwise Failure Crediting (PFC) due to the manner in which it computes diversity. The analogy between implicit fitness sharing and PFC will also be considered.

In the following discussion a failure pattern can be regarded as a string of 0s and 1s i.e.  $\{0, 1\}^n$  where  $n$  is the number of patterns in the training set. We want to measure the diversity with respect to the training set as we want to see on what parts of the training set do the predictors work. In order to measure the diversity with respect to the test set (which basically is the actual definition of diversity as given in [12, 41]), we would similarly need the failure patterns of each individual in the ensemble over the test set. Essentially, a 0 in the failure pattern indicates a predictor failing on the input i.e. gives an incorrect output whereas a 1 means a predictor processing the input correctly.

Given the failure patterns for all the members in the ensemble, the following subsection takes a closer look at how to make use of this information to get the diversity value.

## 5.1. PFC FOR THE ENSEMBLE MEMBERS

Suppose we have two predictors ( $P_1$  and  $P_2$ ) and their failure patterns ( $out_1$  and  $out_2$ ). The hamming distance between the failure patterns of the two predictors would give us a measure of how different these predictors are with respect to each other. The hamming distance does surely give a measure of similarity between two predictors or the output patterns of the two predictors but in order to make this distance useful in computing the diversity in the ensemble (or diversity of the individual member with respect to the ensemble) we calculate the hamming distance between the failure patterns of two individuals and then divide this value by the sum of the number of failures recorded by both individuals (as given by the sum of the number of 0s in the failure patterns of both individuals). In so doing, it can be easily seen that the maximum value for this modified distance calculation is always 1 which can only happen if two predictors make errors on disjoint portions of the training set. Similarly, the minimum value for this distance measure is always 0 which, as is obvious, will happen when the two predictors make errors on exactly the same patterns in the training set or make no errors (which is unlikely in real world problems) at all.

Let this new value in  $[0, 1]$  be called *failure credit*. Now let's consider an ensemble with  $M$  predictors. So we have  $M$  failure pattern vectors  $\{out_1, out_2, \dots, out_M\}$ . We take all pairs of individuals and accumulate the failure credit on each individual i.e. if the failure credit value for a pair of predictors is say  $credit_f$ , then  $credit_f$  will be assigned to both these individuals, and further consideration of pairs would lead to accumulation of  $credit_f$  values on the respective pairs.

Having accumulated the failure credit on each individual in the ensemble, we can now present the measure used to compute the diversity of the individual members with respect to the ensemble i.e. how different a member is with respect to others in the ensemble.

Let the failure credit values accumulated on each individual be  $\{acc\_credit_{f_1}, acc\_credit_{f_2}, \dots, acc\_credit_{f_M}\}$ .

Mathematically, for the  $i$ th member in the ensemble:

$$PFC = \frac{acc\_credit_{f_i}}{M - 1} \quad (14)$$

This measure has a maximum value of 1 and a minimum value of 0 which signifies that a member in the ensemble is very different from others (not having any common failures with any other member in the ensemble) and a member having common and exactly the same failures with respect to all other members in the ensemble respectively. Thus a 0 for one member actually means that all members are similar and all

will have 0 as their PFC value.

**Using PFC.** We propose the use of this new diversity term in place of the NCL penalty function term in our multi-objective formulation.

NCL's penalty term tries to make individual members negatively correlated in their error estimates on the training set i.e. tries to make individuals make errors on different parts of the training set. Minimising this term minimises mutual information between the outputs from two predictors, hence making them different [16]. In case of PFC, we know that if an individual has a PFC value close to 1 then it is considered very different from the rest whereas a PFC value close to 0 signifies similarity. Therefore, the diversity objective in the multi-objective formulation of the problem would change from minimising the NCL penalty function term to maximising the PFC term for each individual in the ensemble.

The multi-objective formulation of the problem would now be as follows:

**Accuracy.** Given a training set with  $N$  patterns. For each network  $k$  in the ensemble,

$$\text{(Minimise) Accuracy}_k = \frac{1}{N} \sum_{i=1}^N (f_k^i - o^i)^2, \quad (15)$$

**Diversity.** Let  $N$  be the number of training patterns and let there be  $M$  members in the ensemble, so for each member  $k$ , the following term (which is the PFC measure) gives an indication of how different it is from other members.

$$\text{(Maximise) Diversity}_k = \frac{acc\_credit_{f_k}}{M - 1}, \quad (16)$$

where  $acc\_credit_{f_k}$  is the accumulated failure credit value (computing this value has been discussed earlier on) on individual  $k$ .

## 5.2. ANALOGY WITH IMPLICIT FITNESS SHARING

The seminal work on implicit fitness sharing was done by Smith et al. [42] who described it from an immune system perspective where the immune system was considered as a simple evolutionary learning system. Much work on implicit fitness sharing has been done since then [18, 20, 21, 22]. The basic idea behind implicit fitness sharing is that it tries to maintain the diversity in the population being evolved by a competitive co-evolutionary fitness manipulation mechanism. Darwin and Yao [20] used implicit fitness sharing to evolve game playing

strategies (a strategy can be represented as a neural network). They evolve strategies to play the 2 player iterated players dilemma game using speciation (by implicit fitness sharing), which generates diverse experts from within the search space as potential solutions. These solution strategies are then combined, resulting in the combination performing better against unseen opponent strategies when compared with solution strategies resulting from simple evolution playing against these opponents. Speciation is achieved by a fitness manipulation procedure. Strategies play with each other, switching their roles between test strategies and being a member of the population being evolved (test strategies are equivalent to antigens and the population being evolved models the antibody population from the immune system [42] point of view). Each strategy within the population is considered as a test strategy at least  $C$  times and a fitness manipulation process carried out in every iteration. Essentially, a sample (of some size  $\sigma$ ) of strategies is selected which play against the test strategy one by one. A credit/payoff is assigned to the strategy which performs best (or strategy having the largest winning margin) with respect to the test strategy. The above discussion follows [20].

The way in which failure credits are being assigned to the individuals in PFC does bear a resemblance to implicit fitness sharing. The only difference is that we compare all pairs of individuals here, whereas in implicit fitness sharing we pick a sample of individuals randomly and compare this sample with some test individual but we do this many times which may lead to a similar credit/payoff assignment. If the sample size (in implicit fitness sharing) is 1 individual then the payoff assignment is quite similar to how we assign credits to calculate the diversity. If an individual is very different i.e. has a large winning margin with respect to a test individual or has a high score (the score meaning hamming distance between the two individuals) then it will receive payoff (which in the original implementation [42] is the score). On the other hand, if it is similar (or the hamming distance is zero i.e. the score is zero), it would not receive any, which is sort of similar to what this diversity calculation method does. The payoff for each individual that emerges might in a way be tantamount to the credits accumulated using the all pairs strategy for diversity calculation in PFC. Instead of putting two strategies against each other and ending up with a score as in [20], two individuals in our case play by simply determining how different they are on the training set. Hence, this credit assignment method for diversity calculation in PFC can be considered as a *constrained version of implicit fitness sharing*. In effect, contribution to the credits accumulated on an individual by other individuals is more if it is different from others in the population.

This analogy essentially shows us a remarkable connection between machine learning (wanting diverse members in the ensemble) and evolutionary computation (wanting species in a population) and tells us that certain aspects of both fields when moulded into one unified approach, can lead to promising ideas.

Another very valid point in using this diversity measure is that fitness sharing has actually been put to good use in multi-objective evolutionary algorithms [23, 24, 25, 29, 43]. It tries to make individuals in the pareto set as diverse as possible. This is a much required property in multi-objective optimisation algorithms because we are concerned with getting a set of near optimal solutions which are well distributed on the pareto front. This fact is also true for ensembles and since we are using multi-objective evolutionary optimisation, utilising a measure such as PFC seems a good proposition.

Moreover, implicit fitness sharing has a theoretical base (in it being similar to fitness sharing [19, 31, 32] as shown in [42]). This gives theoretical validity to our diversity measure i.e. PFC, which we will consider deriving in future. In this paper however, the working capability of PFC is expressed by some empirical results considered in the following section.

## 6. Results and Discussion

This section presents some results obtained on testing DIVACE on 2 benchmark data sets (Australian credit card assessment dataset and Diabetes dataset), available by anonymous ftp from ice.uci.edu in /pub/machine-learning-databases [9].

The Australian credit card assessment dataset contains information needed to assess (classify) credit card applications on the basis of a set of attributes. In all, it has 690 instances or patterns with 14 attributes and 1 class attribute. Out of the 14 attributes which are used as the input pattern to the ensemble, 6 are numeric/continuous and 8 are categorical/discrete. This dataset is interesting in that there is a good mix of continuous and discrete attributes and the discrete attributes have from 2 to 14 possible values. Moreover, there are some missing values too which have been replaced either by the mode of the attribute (in case of discrete attributes) or the mean (in case of continuous attributes). The predicted class is binary where 1 means awarding credit and a 0 means not. The class distribution is pretty uniform in that 307 patterns belong to one class and 383 belong to the other i.e. 44.5% of the dataset conforms to one class and 55.5% to the other.

The Diabetes dataset again has two classes where the binary-valued variable investigated is whether the patient shows signs of diabetes given some attribute values. This dataset has 768 patterns with 500 belonging to one class and 268 to the other. This dataset is supposed to be rather difficult [36, 5] to classify. In all, each pattern has 8 attributes and 1 class variable. All the attributes are continuous. Class value 1 indicates a positive test for diabetes and class 0 means negative.

The experimental setup is similar to that in [36, 3, 5] in order to facilitate comparison with previous work and for consistency.

*For the Australian credit card assessment dataset:*

We used n-fold cross validation and divided the dataset into 10 mutually exclusive folds where the class distribution was maintained in each fold. We used 9 out of the 10 folds for training and the remaining fold was used for testing. After every generation, we recorded the training and test accuracies which are simply the ratios of the number of patterns classified correctly to the total number of patterns within the training and test sets respectively. We finally calculated the classification accuracy rate by averaging the final training and test accuracies (i.e. training and test accuracies after the final generation in the evolutionary process) over all the training-testing sessions (in this case 10 as there were 10 folds).

*For the Diabetes dataset:*

The whole testing scenario was similar to that used in case of the Australian credit card assessment dataset with one difference. Instead of 10 folds for the n-fold cross validation process, we use 12 folds in this case. Thus 11 out of the 12 folds were used for training and the remaining fold was used as as test set. Here again, the training and testing accuracies as well as the final classification accuracy were calculated.

Table I shows the parameters used in our experiments. Another aspect of the experimental setup worth mentioning here is the way the individual networks in the population are combined to form an ensemble. The way in which we combine the outputs of the network is very important as it can tell which combination rule works better under the given situation and can help compare results with previous approaches. There are mainly 3 types of combination rules that we have considered in our experiments and which were considered in [36, 3, 5]. These are: simple averaging, majority voting and winner-takes-all.

Table I. Parameters used in DIVACE.

Parameter	Value
Population size	25
# generations	200
# training epochs	5
Learning rate for BP	0.003
# hidden units in each network within the population	5
# folds used in Australian Credit Card Assessment problem	10
# folds used in Diabetes problem	12
# = number of	

**Simple averaging:** we simple take the average of the outputs of the ensemble members.

**Majority voting:** the outputs of individual networks are considered in a way such that we count the number of individuals that predict each class and then the class which has the majority vote (i.e. the class on which the individual networks agreed on most) is set as the output of the ensemble.

**Winner-takes-all:** in this case, the network with the highest/lowest activation is considered as the representative for the whole ensemble and its outputs are used as the output of the ensemble.

We will also present some statistical analysis on the results we get from our experiments using confidence intervals. A confidence interval provides statistical backing to a sample estimate - estimate such as mean of a set of values (sample) where the sample comes from a population of values - telling how well the sample relates to the entire population of values. It is a range of values determined from the sample set telling the probability (confidence level) that the values from the population will lie within this interval (range).

In our case, the values on which we are going to establish confidence levels (and hence confidence intervals) are the training and testing accuracies of the ensemble after the final generation from all the runs of the algorithm (runs signifies runs on different folds here). Thus, with n-fold cross validation, our sample set has  $n$  (for Australian credit

card assessment  $n = 10$  and for Diabetes  $n = 12$ ) values. We need to establish how well the sample mean (mean of the  $n$  values) estimates the mean of the entire population. To get the true mean we would need to have the whole population of values which, in essence, has infinite elements and is practically impossible to realise. Hence, the use of confidence intervals on our experimental results can establish a good statistical backing i.e. can quantify how good the results (and the algorithm) are.

### 6.1. ORIGINAL DIVACE

During the course of the evolutionary process, it was expected that each member in the Pareto (non-dominated) set (after every generation) would perform well on different parts of the training set. Table II shows the performance accuracy of the formed ensemble on the Australian credit card assessment dataset. Table III shows the same for the Diabetes dataset.

As far as the confidence intervals for the mean training and testing accuracies are concerned, we can see from tables IV and V that the mean training accuracy for all the three combination rules (viz. simple averaging, majority voting and winner-takes-all) has a lot less variance as compared with the same for the testing accuracy. This is mainly because we take the whole training set with consistent class distributions for each network in the population to train on, which means that the networks fit the training set quite well.

Table VI shows the results from the second formulation of MPANN given in [3] which is one of the many algorithms we use to compare our algorithm with. We chose this algorithm for comparison purposes separately here because it is a multi-objective evolutionary ensemble construction technique, as a result, giving a fairer comparison. In addition, the training setup that we use for DIVACE is identical to that of this algorithm. By training setup we mean that with DIVACE, we use the whole training set in conjunction with the two objective function evaluations and this is exactly the case in the second formulation of MPANN where the two objectives are quadratic error functions but one of these has a noise component added to it. Comparison with other approaches is considered later on in this section.

From the tables giving confidence intervals for both MPANN (tables VII and VIII) and DIVACE (tables IV and V), it is seen that DIVACE performs very well on the Australian credit card assessment dataset as is evident from the confidence intervals established for both training and testing (for all three combination rules). This essentially suggests that DIVACE is competitive. However, in case of the Diabetes

Table II. Performance (accuracy rates) of the ensemble formed using DIVACE on the Australian credit card assessment dataset.

Australian credit card assessment dataset						
	Simple Averaging		Majority Voting		Winner-takes-all	
	Training	Testing	Training	Testing	Training	Testing
Mean	0.872	0.862	0.867	0.857	0.855	0.849
SD	0.007	0.049	0.007	0.049	0.007	0.053
Max	0.884	0.927	0.879	0.927	0.864	0.927
Min	0.859	0.753	0.856	0.768	0.842	0.753

dataset, although DIVACE performs very well as far as the training is concerned and here too for all combination rules, the confidence intervals established for the estimated mean are marginally greater than those exhibited by MPANN (but still the testing accuracy estimated is better; it is just marginally more variable i.e.  $\pm 0.0105 \approx \pm 1\%$ ). The value  $\pm 0.0105$  is how different the confidence intervals in both MPANN and DIVACE are on an average over all combination rules. The Diabetes dataset is difficult to classify as established by various researchers and as stated in [36, 5] which could imply that some areas within DIVACE need to be tweaked so as to cater for this incongruity. However, the results are competitive. An important point here is that the lower limits on the means acceptable with a confidence of 95% due to the confidence intervals established (using all three combination rules) for the Diabetes dataset on the testing accuracies are higher in DIVACE as compared to MPANN. The lower limits for DIVACE in this case are  $\{0.745, 0.734, 0.738\}$  whereas for MPANN, these are  $\{0.725, 0.725, 0.728\}$ . This fact implies that although the variability/variance in DIVACE as to the testing accuracy on different folds is greater (making the confidence interval larger), the confidence intervals start at a higher point (i.e. are shifted upward) as compared to MPANN. In other words, *for DIVACE, the chance that the mean estimated lies within a smaller interval is low but it is a fact that this interval or the lower limits for the acceptable means to start at are higher than the lower limits of MPANN.* The fact that the training accuracy is also a lot better (testing accuracy being better but having a marginally lower confidence level yet with higher lower limits for the acceptable mean) says that the algorithm indeed is competitive with MPANN.

Table III. Performance (accuracy rates) of the ensemble formed using DIVACE on the Diabetes dataset.

Diabetes dataset						
	Simple Averaging		Majority Voting		Winner-takes-all	
	Training	Testing	Training	Testing	Training	Testing
Mean	0.780	0.773	0.783	0.766	0.766	0.766
SD	0.006	0.050	0.005	0.057	0.017	0.049
Max	0.791	0.859	0.791	0.875	0.796	0.843
Min	0.768	0.687	0.772	0.671	0.730	0.671

Table IV. Confidence intervals with a confidence level of 95% for training and testing of DIVACE on the Australian credit card assessment dataset.

Combination rule	Training	Testing
Simple Averaging	$.872 \pm .0043$	$.862 \pm .0303$
Majority Voting	$.867 \pm .0043$	$.857 \pm .0303$
Winner-takes-all	$.855 \pm .0043$	$.849 \pm .0328$

Table V. Confidence intervals with a confidence level of 95% for training and testing of DIVACE on the Diabetes dataset.

Combination rule	Training	Testing
Simple Averaging	$.780 \pm .0033$	$.773 \pm .0282$
Majority Voting	$.783 \pm .0028$	$.766 \pm .0322$
Winner-takes-all	$.766 \pm .0096$	$.766 \pm .0277$

Table VI. Performance (accuracy rates) of the ensemble formed using the second formulation of MPANN [3] on the Australian credit card assessment and the Diabetes datasets.

Australian credit card assessment dataset						
	Simple Averaging		Majority Voting		Winner-takes-all	
	Training	Testing	Training	Testing	Training	Testing
Mean	0.850	0.844	0.852	0.844	0.834	0.824
SD	0.017	0.058	0.015	0.056	0.027	0.053
Max	0.869	0.913	0.871	0.913	0.866	0.9
Min	0.809	0.724	0.821	0.724	0.768	0.724
Diabetes dataset						
	Simple Averaging		Majority Voting		Winner-takes-all	
	Training	Testing	Training	Testing	Training	Testing
Mean	0.753	0.744	0.755	0.744	0.753	0.746
SD	0.019	0.034	0.019	0.034	0.020	0.032
Max	0.795	0.806	0.795	0.806	0.795	0.802
Min	0.738	0.690	0.738	0.690	0.738	0.690

Table VII. Confidence intervals with a confidence level of 95% for training and testing of MPANN on the Australian credit card assessment dataset.

Combination rule	Training	Testing
Simple Averaging	.850 ± .0105	.844 ± .0359
Majority Voting	.852 ± .0092	.844 ± .0347
Winner-takes-all	.834 ± .0167	.824 ± .0328

Table VIII. Confidence intervals with a confidence level of 95% for training and testing of MPANN on the Diabetes dataset.

Combination rule	Training	Testing
Simple Averaging	.753 ± .0107	.744 ± .0192
Majority Voting	.755 ± .0107	.744 ± .0192
Winner-takes-all	.753 ± .0113	.746 ± .0181

## 6.2. DIVACE WITH PFC

We try to establish two results here. While testing the utility of the PFC measure, we also try to show that DIVACE stays competitive with modifications at the objective function level. In order to do this, DIVACE was tested on the previously mentioned datasets (Australian credit card assessment and Diabetes) using PFC as its diversity objective. Tables IX and XI show the performance accuracies of DIVACE (using PFC) under various combination rules. The trend, as far as the mean and standard deviations for both training and testing are concerned, is still similar to DIVACE with the original problem formulation.

Given that PFC directly acts on the failures that each individual makes on the training set and not on the probabilities of failures (as in the case of the NCL penalty function term), this makes diversity calculation somewhat quantised or less continuous which makes the accuracy objective have a greater influence resulting in a lower test accuracy as compared to the rates obtained using the original problem formulation. However, a closer look at results from the winner takes all combination rule reveals that the performance with respect to both training and testing is much better indicating that the members in the ensemble, as the same time as being very accurate, are also doing very well on the test data. This could be attributed to the fact that we do not consider the ensemble output while calculating diversity for each individual, rather, we only calculate individual outputs. Doing this makes the individuals uncorrelated to all other members explicitly as opposed to minimising the correlation between outputs of networks and the ensemble output (which in the winner-takes-all case is the output from just one network) as with the NCL penalty term. Hence, instead of making all networks different from just one network, we try to make it different from all others in the population using PFC and so we get the test accuracy improvement, as evident from tables IX and XI.

Similarly, the confidence levels established by DIVACE using PFC on both Australian credit dataset (Table X) and Diabetes dataset (Table XII) show that DIVACE with PFC performs better than MPANN on the training front and that it is very competitive on the testing front with lower levels of the means established (for all three combination rules) being  $\{0.722, 0.717, 0.748\}$  as opposed to  $\{0.725, 0.725, 0.728\}$  for MPANN (Table VIII) on the Diabetes dataset. The same for the Australian credit dataset are  $\{0.828, 0.822, 0.847\}$  which are better than those obtained for MPANN i.e.  $\{0.808, 0.809, 0.791\}$  (Table VII).

Table IX. Performance (accuracy rates) of the ensemble formed using DIVACE with PFC on the Australian credit card assessment dataset.

Australian credit card assessment dataset						
	Simple Averaging		Majority Voting		Winner-takes-all	
	Training	Testing	Training	Testing	Training	Testing
Mean	0.872	0.862	0.870	0.855	0.891	0.875
SD	0.011	0.053	0.007	0.053	0.025	0.044
Max	0.893	0.927	0.884	0.927	0.962	0.942
Min	0.856	0.753	0.859	0.753	0.872	0.797

Table X. Confidence intervals with a confidence level of 95% for training and testing of DIVACE with PFC on the Australian credit card assessment dataset.

Combination rule	Training	Testing
Simple Averaging	$.872 \pm .0068$	$.862 \pm .0331$
Majority Voting	$.870 \pm .0046$	$.855 \pm .0330$
Winner-takes-all	$.891 \pm .0160$	$.875 \pm .0278$

Table XI. Performance (accuracy rates) of the ensemble formed using DIVACE with PFC on the Diabetes dataset.

Diabetes dataset						
	Simple Averaging		Majority Voting		Winner-takes-all	
	Training	Testing	Training	Testing	Training	Testing
Mean	0.781	0.759	0.770	0.747	0.782	0.774
SD	0.009	0.064	0.027	0.051	0.006	0.045
Max	0.798	0.843	0.791	0.812	0.791	0.843
Min	0.758	0.656	0.691	0.640	0.769	0.703

Table XII. Confidence intervals with a confidence level of 95% for training and testing of DIVACE with PFC on the Diabetes dataset.

Combination rule	Training	Testing
Simple Averaging	$.781 \pm .0055$	$.759 \pm .0362$
Majority Voting	$.770 \pm .0153$	$.747 \pm .0294$
Winner-takes-all	$.782 \pm .0034$	$.774 \pm .0259$

### 6.3. COMPARISON WITH OTHER WORK

Here, we compare DIVACE with a variety of learning algorithms, not necessarily evolutionary nor producing ensembles (mainly due to lack of such results), yet giving a reasonable idea as to where DIVACE, as a learning algorithm, stands on a wider scale.

Tables XIII and XIV show the average test error rates of learners generated by various algorithms. The algorithms used for comparison can be categorised into five classes: evolutionary ensemble methods (MPANN [3], EENCL [36]), statistical methods (Discrim, Quadisc, Logdisc, SMART, ALLOC80, k-NN, CASTLE, NaiveBay), decision trees based methods (CART, IndCART, NewID,  $AC^2$ , Baytree, Cal5, C4.5), rule based methods (CN2, ITrule) and neural network based methods (BP, LVQ, RBF, DIPOL92). Details of the algorithms in the latter four classes can be found in [37]. The error rates refer to the percentage of wrong classifications on the test set. The rates chosen for comparison, in case of DIVACE, are those from the simple averaging combination rule and the winner-takes-all rule respectively, for the two formulations of DIVACE considered. As can be seen, both variants of DIVACE were able to achieve competitive and mostly better generalisation performance when compared with the best of the other algorithms tested for both datasets.

Table XIII. Comparison of DIVACE with other learning algorithms from [37] in terms of the average test error rates for the Australian credit card assessment dataset. Results are averaged on 10-fold cross validation. The two rates, as indicated for DIVACE, are for DIVACE with different diversity measures (original formulation and PFC respectively).

Algorithm	Error Rate	Algorithm	Error Rate
DIVACE	0.138, 0.125	NewID	0.181
MPANN	0.156	$AC^2$	0.181
EENCL	0.135	Baytree	0.171
Discrim	0.141	NaiveBay	0.151
Quadisc	0.207	CN2	0.204
Logdisc	0.141	C4.5	0.155
SMART	0.158	ITrule	0.137
ALLOC80	0.201	Cal5	0.131
k-NN	0.181	DIPOL92	0.141
CASTLE	0.148	BP	0.154
CART	0.145	RBF	0.145
IndCART	0.152	LVQ	0.197

Table XIV. Comparison of DIVACE with other learning algorithms from [37] in terms of the average test error rates for the Diabetes dataset. Results are averaged on 12-fold cross validation. The two rates, as indicated for DIVACE, are for DIVACE with different diversity measures (original formulation and PFC respectively).

Algorithm	Error Rate	Algorithm	Error Rate
DIVACE	0.227, 0.226	NewID	0.289
MPANN	0.254	$AC^2$	0.276
EENCL	0.221	Baytree	0.271
Discrim	0.225	NaiveBay	0.262
Quadisc	0.262	CN2	0.289
Logdisc	0.223	C4.5	0.27
SMART	0.232	ITrule	0.245
ALLOC80	0.301	Cal5	0.25
k-NN	0.324	DIPOL92	0.224
CASTLE	0.258	BP	0.248
CART	0.255	RBF	0.243
IndCART	0.271	LVQ	0.272

## 7. Conclusion

DIVACE tackles the ensemble learning problem within a multi-objective evolutionary setup with the two objectives being diversity and accuracy which are two factors that are in conflict and much required within an ensemble in order that it generalises well or at least generalises better than the individual members that it is made of. The explicit use of diversity and accuracy in the formulation of the problem tries to make the evolutionary process search for a good trade-off between these two factors. Explicit use of diversity and accuracy has been taken up in the evolutionary computation literature as well [6] and has been shown to be a good way of enforcing diversity in the population as also seen here. Empirical studies on DIVACE and its variant are considered here and it is shown that DIVACE does result in ensembles that have diverse and accurate members resulting in good performance when compared with previously studied approaches. Statistical analysis of the results show that DIVACE is a promising algorithm. We also establish that the basic idea behind DIVACE is much more flexible than what the original formulation of the algorithm exhibits by considering a different multi-objective formulation i.e. using a different diversity measure (PFC) and seeing that the algorithm remains competitive. This also results in showing the viability of PFC.

Since the issue at hand is the achievement of a good trade-off between diversity and accuracy, we can try and enforce diversity by considering ensembles which have members trained using disparate learning methodologies (hybrid ensembles) instead of simply using feed forward neural networks trained using BP. Speciation/fitness sharing can explicitly be employed in order to enforce diversity as well. Moreover, various ensemble methods viz. bagging, boosting etc. could be combined [26] and we could, for instance, make specialised crossover operators which follow the ideas behind bagging and boosting in that, offspring could be generated by establishing a probability distribution over the training set conforming to the errors made by the non-dominated individuals (parents), and this probability distribution used in order to generate a training set for the child to train on. These are some future directions for research worth looking into and exploring the conceptual robustness of DIVACE. As a first step towards this goal, we recently proposed an evolutionary framework for the construction of diverse hybrid ensembles which essentially provides us with a methodology using which one can generate multi-objective evolutionary ensemble construction algorithms at multiple levels of abstraction. More on this framework can be found in [17]. DIVACE, in this case, plays the role of managing the trade-off between diversity and accuracy at various levels.

Initial experiments [17] show that the framework is indeed promising, again showing the promise of DIVACE.

## References

1. H. A. Abbass. A memetic pareto evolutionary approach to artificial neural networks. In *Proceedings of the 14th Australian Joint Conference on Artificial Intelligence*, pages 1–12, Berlin, 2000. Springer-Verlag.
2. H. A. Abbass. The self-adaptive pareto differential evolution algorithm. In D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 831–836. IEEE Press, 2002.
3. H. A. Abbass. Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization. In *The IEEE 2003 Conference on Evolutionary Computation*, volume 3, pages 2074–2080. IEEE Press, 2003.
4. H. A. Abbass. Speeding up backpropagation using multiobjective evolutionary algorithms. *Neural Computation*, 15(11):2705–2726, November 2003.
5. H. A. Abbass. Pareto neuro-ensemble. In *16th Australian Joint Conference on Artificial Intelligence*, pages 554–566, Perth, Australia, 2003a. Springer.
6. H. A. Abbass and K. Deb. Searching under multi-evolutionary pressures. In *Proceedings of the 2003 Evolutionary Multiobjective Optimization Conference (EMO03), LNCS-2632*, pages 391–404, Berlin, 2003. Springer-Verlag.
7. H. A. Abbass, R. Sarker, and C. Newton. Pde: A pareto-frontier differential evolution approach for multi-objective optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2001)*, volume 2, pages 971–978. IEEE Press, 2001.
8. J. Baldwin. A new factor in evolution. *American Naturalist*, 30:441–451, 1896.
9. C. Blake and C. Merz. UCI repository of machine learning databases, 1998.
10. E. Boers, M. Borst, and I. Sprinkhuizen-Kuyper. Evolving artificial neural networks using the “baldwin effect”. Technical Report 95-14, Leiden University, Department of Computer Science, The Netherlands, 1995.
11. G. Brown. *Diversity in Neural Network Ensembles*. PhD thesis, School of Computer Science, University of Birmingham, 2004.
12. G. Brown, J. Wyatt, R. Harris, and X. Yao. Diversity creation methods: A survey and categorisation. *Journal of Information Fusion (to appear)*, 2004.
13. G. Brown and J. L. Wyatt. Negative correlation learning and the ambiguity family of ensemble methods. In *Proc. Int. Workshop on Multiple Classifier Systems (LNCS 2709)*, pages 266–275, Guildford, Surrey, June 2003. Springer.
14. G. Brown and J. L. Wyatt. The use of the ambiguity decomposition in neural network ensemble learning methods. In T. Fawcett and N. Mishra, editors, *20th International Conference on Machine Learning (ICML’03)*, Washington DC, USA, August 2003.
15. A. Chandra. Evolutionary approach to tackling the trade-off between diversity and accuracy in neural network ensembles. Technical report, School of Computer Science, The University of Birmingham, UK, 2004.
16. A. Chandra and X. Yao. DIVACE: Diverse and Accurate Ensemble Learning Algorithm. In *Proc. 5th Intl. Conference on Intelligent Data Engineering and Automated Learning (LNCS 3177)*, pages 619–625, Exeter, UK, August 2004. Springer-Verlag.

17. A. Chandra and X. Yao. Evolutionary framework for the construction of diverse hybrid ensembles. In *Proc. 13th European Symposium on Artificial Neural Networks*, pages 253–258, Brugge, Belgium, April 2005. d-side.
18. P. J. Darwen. *Co-Evolutionary Learning by Automatic Modularisation with Speciation*. PhD thesis, University of New South Wales, November 1996.
19. P. J. Darwen and X. Yao. A Dilemma for Fitness Sharing with a Scaling Function. In *Proceedings of the Second IEEE International Conference on Evolutionary Computation*, Piscataway, New Jersey, 1995. IEEE Press.
20. P. J. Darwen and X. Yao. Automatic modularization by speciation. In *IEEE International Conference on Evolutionary Computation*, pages 88–93. IEEE Press, May 1996.
21. P. J. Darwen and X. Yao. Every niching method has its niche: Fitness sharing and implicit sharing compared. In *Proc. of the 4th International Conference on Parallel Problem Solving from Nature (PPSN-IV), (LNCS-1141)*, pages 398–407, Berlin, September 1996. Springer-Verlag.
22. P. J. Darwen and X. Yao. Speciation as automatic categorical modularization. *IEEE Trans. on Evolutionary Computation*, 1(2):100–108, 1997.
23. K. Deb. Multi-objective evolutionary algorithms: Introducing bias among pareto-optimal solutions. Technical Report 99002, Kanpur Genetic Algorithm Group, Department of Mechanical Engineering, Indian Institute of Technology, Kanpur, India, 1999.
24. K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Chichester, UK : Wiley, 2001.
25. K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.
26. T. G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.
27. S. Forrest, R. E. Smith, B. Javornik, and A. S. Perelson. Using genetic algorithms to explore pattern recognition in the immune system. *Evolutionary Computation*, 1(3):191–211, 1993.
28. L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.
29. J. Horn, N. Nafpliotis, and D. E. Goldberg. A Niche Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, New Jersey, 1994. IEEE Service Center.
30. M. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transactions on Neural Networks*, 14(4):820–834, July 2003.
31. V. Khare and X. Yao. Artificial Speciation and Automatic Modularisation. In L. Wang, K. C. Tan, T. Furuhashi, J.-H. Kim, and X. Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02)*, number 1, pages 56–60, Singapore, November 2002.
32. V. Khare and X. Yao. Artificial Speciation of Neural Network Ensembles. In J. A. Bullinaria, editor, *Proc. of the 2002 UK Workshop on Computational Intelligence (UKCI'02)*, pages 96–103, Birmingham, September 2002.

33. A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. *NIPS*, 7:231–238, 1995.
34. W. B. Langdon, S. J. Barrett, and B. F. Buxton. Combining decision trees and neural networks for drug discovery. In *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, pages 60–70, Kinsale, Ireland, 3-5 April 2002.
35. Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.
36. Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE-EC*, 4(4):380, November 2000.
37. D. Michie, D. Spiegelhalter, and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited, 1994.
38. D. Opitz. Feature selection for ensembles. In *Proceedings of 16th National Conference on Artificial Intelligence (AAAI)*, pages 379–384, 1999.
39. D. Opitz and R. Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.
40. D. W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural-network ensemble. *NIPS*, 8:535–541, 1996.
41. A. Sharkey. *Multi-Net Systems*, chapter Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems, pages 1–30. Springer-Verlag, 1999.
42. R. Smith, S. Forrest, and A. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1993.
43. N. Srinivas and K. Deb. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.
44. K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.
45. R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, International Computer Science Institute, Berkeley, USA, 1995.
46. K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, February 1996.
47. X. Yao. Evolving artificial neural networks. In *Proceedings of the IEEE*, volume 87, pages 1423–1447. IEEE, September 1999.

