Competitive co-evolution of multi-layer perceptron classifiers

Castellani, Marco

DOI:
10.1007/s00500-017-2587-6

Soft Comput
DOI 10.1007/s00500-017-2587-6

METHODOLOGIES AND APPLICATION

Competitive co-evolution of multi-layer perceptron classifiers

Marco Castellani[1]

Abstract This paper analyses the competitive approach to the co-evolutionary training of multi-layer perceptron classifiers. Two algorithms were tested: the first opposes a population of classifiers to a population of training patterns; the second pits a population of classifiers against a population of subsets of training patterns. The classifiers are regarded as predators that need to 'capture' (correctly categorise) the prey (training patterns). Success for the predators is measured on their ability to capture prey. Success for the prey is measured on their ability to escape predation (be misclassified). The aim of the procedure is to create an evolutionary tug-of-war between the best classifiers and the most difficult data samples, increasing the efficiency and accuracy of the learning process. The two co-evolutionary algorithms were tested on a number of well-known benchmarks and on several artificial data sets modelling different kinds of common classification problems such as overlapping data categories, noisy training inputs, and unbalanced data classes. The performance of the co-evolutionary methods was compared with that of two traditional training techniques: the standard backpropagation rule and a conventional evolutionary algorithm. The co-evolutionary procedures achieved top accuracy in all classification problems. They particularly excelled on data sets containing noisy training inputs, where they outperformed the backpropagation rule, and on tasks involving unbalanced data classes, where they outperformed both backpropagation and the conventional evolutionary algorithm. Compared to the standard evolutionary algorithm, the co-evolutionary procedures were able to obtain similar or superior learning accuracies, whilst needing considerably less presentations of the training patterns. This economy in the use of training patterns translated into significant savings in computational overheads and algorithms running time.

Keywords Evolutionary algorithms · Co-evolution · Predator–prey systems · Competition · Multi-layer perceptron · Learning

Communicated by V. Loia.

✉ Marco Castellani
m.castellani@bham.ac.uk

[1] School of Engineering, University of Birmingham, Birmingham B15 2TT, UK

1 Introduction

The multi-layer perceptron (MLP) (Haykin 2009) is to date one of the most popular and versatile artificial neural network (ANN) models. Thanks to its capability of mapping arbitrarily complex decision regions (Lippmann 1987), the MLP has been extensively used for pattern classification. This kind of ANN is trained to perform the desired categorisation task via the repeated presentation of a set of instances of the various classes. During the training process, the weights of the neuron connections are adjusted so as to reduce the network's classification error.

Training an MLP is essentially a search problem in the space of the ANN weights, where the solution is the configuration that minimises the overall classification error. Standard gradient-based training algorithms like the backpropagation (BP) rule (Rumelhart and McClelland 1986) are liable to converge to suboptimal peaks of accuracy. For this reason, evolutionary algorithms (EAs) (Fogel 2000) have been extensively used to design and train MLP classifiers (Yao 1999; Azzini and Tettamanzi 2011; Castellani 2013).

The main advantage of EAs is their global search approach, which allows them to escape suboptimal weight configurations. The main drawback is the computational

Published online: 04 April 2017

complexity of the procedure, which becomes problematic if large ANNs and training sets are involved. Hence, the standard BP rule is often used to speed up the learning process (Lamarckian evolution Aboitiz 1992). Yet, despite its ability to descend quickly the ANN error surface, BP is computationally demanding and in some cases the overall training process may be extremely lengthy.

One of the most computationally intensive tasks in an evolutionary MLP training procedure is the evaluation of the candidate solutions, which entails the presentation of the whole training set of examples to every individual. Namely, each of the M training patterns must be forward processed by each of the N solutions. If Lamarckian learning is used, a further M forward and M backward propagations are added. Randomly sampling only a proportion of the training data at each evolution cycle (Castellani 2013) alleviates the time complexity of the EA, but may degrade the learning accuracy and speed.

Typical MLP evolutionary learning curves are characterised by an exponentially decreasing behaviour, with an initial rapid drop of the average classification error and a long tail of slower improvement. That is, the solutions learn quickly to classify the bulk of the examples and spend the rest of the time learning to categorise the most 'difficult' cases (Paredis 1995). Yet, in standard EAs the entire training set is used throughout the whole procedure. If the training process progressively focused on the most problematic examples, the time complexity of the algorithm would be greatly reduced.

Paredis (1995) devised a co-evolutionary genetic algorithm (CGA) mimicking predator–prey interaction, where a population of MLP classifiers is opposed to a training set of patterns. Classifiers (predators) and individual training patterns (prey) are pitted one against the other in one-to-one tournaments. The fitness of the predators is evaluated on their capability to 'capture' (classify correctly) the prey. The fitness of the prey is evaluated on their ability to escape predation (i.e. how often it is misclassified by the MLPs). Since the likelihood that a prey is selected for a tournament is based on its fitness (i.e. difficulty), the learning procedure focuses mostly on unsolved cases. In this way, CGA is able to reduce considerably the computational effort.

Paredis proved the effectiveness and efficiency of co-evolutionary ANN training (Paredis 1995). However, despite the success and simplicity of CGA, competitive predator–prey systems have seldom been used for ANN training. Indeed, to date most co-evolutionary methods are based on cooperative procedures, where separate populations of components learn to collaborate like symbiotic organisms to solve a task (Potter and Jong 2000; García-Pedrajas et al. 2003; Gomez et al. 2008; Chandra 2013; Rivera et al. 2013).

This study investigates the effectiveness and efficacy of the competitive approach to MLP classifier evolution, testing two alternative methods on problems of different nature. The literature on the subject is reviewed in Sect. 2. Aims and objectives of the paper are stated in Sect. 3. The two predator–prey algorithms are presented in Sect. 4, whilst the experimental set-up is outlined in Sect. 5. Section 6 presents the experimental results of the tests. Section 7 investigates the use of predator–prey procedures on unbalanced data sets. Section 8 discusses the experimental results, and Sect. 9 concludes the paper.

2 Predator–prey algorithms

Popovici et al. (2012) divided competitive co-evolutionary algorithms into single- and multi-population procedures. The first group pitches individuals belonging to the same population against each other to evaluate their ability to perform a given task. Typical examples are the evolution of game players (tic-tac-toe Angeline and Pollack 1993; backgammon Pollack and Blair 1998; checkers Fogel 2002) or robots (football teams Uchibe and Asada 2006), where one individual or team is evaluated on the outcome of one or more games against one or more opponents.

The second group includes procedures featuring at least two populations that co-evolve separately and found application mostly in Artificial Life (Ray 1991; Nolfi and Floreano 1998; Rajagopalan et al. 2010; Nolfi 2012). Multi-population methods are often modelled on predator–prey (sometimes referred to as host–parasite) systems. Since biological co-evolution is defined as the reciprocal influence of two (or more) species on each other's evolution (Popovici et al. 2012), some authors regard as 'co-evolutionary' only multi-population procedures. Henceforth, unless explicitly stated, the terms co-evolutionary and competitive will be used exclusively to refer to multi-population predator–prey algorithms.

Predator–prey interactions were recognised by Darwin as a major factor in evolutionary change. Mitchell et al. (2006) pointed out three main factors that determine the success of competitive co-evolution: the mutual interactions that constantly change the fitness landscape faced by each species, the fostering of population diversity, and the evolutionary arms races that develop between predators and prey. In detail, predators and prey are always changing to target each other's weaknesses. In doing so, they continuously improve their abilities in a never ending struggle for supremacy. They also constantly modify the fitness landscape for the other species, preventing its premature convergence and stagnation in local fitness optima (Floreano and Nolfi 1997).

The primary scope of competitive co-evolution is to foster a successful arms race between opposed populations of problems and solvers (de Boer and Hogeweg 2012). The first example of co-evolutionary algorithm is due to Hillis (1990), who simulated a host–parasite system to generate minimal sorting networks. In his work, a population of sorting networks (the hosts, prey) competes against a population of

training problems, that is, sets of test cases (the parasites, predators). In the evaluation phase, each host is paired to a parasite, and the fitness of the first (how many test cases it solves) is complementary to the fitness of the second (how many test cases the host fails). Complementary fitness is a fundamental element to promote competition amongst species; the success of one is the failure of the other.

2.1 Predator–prey training of ANNs

In his co-evolutionary CGA, Paredis (1995) employs the GENITOR (Whitley 1989) genetic algorithm to evolve the predators (MLPs). The prey (training samples) do not undergo evolution. However, their fitness is updated throughout the algorithm and determines the frequency each prey interacts with the predators. Predator–prey interactions take place in one-to-one tournaments where a pattern is paired to a classifier. The purpose of the tournaments is to update the fitness of the individuals. The classifier is awarded a win if it categorises correctly the training pattern; otherwise, the latter is the winner (complementary fitness). The fitness of a given predator or prey sums up the number of wins in the last 20 tournaments attended (lifetime fitness evaluation).

The prey are picked for tournaments using a stochastic selection procedure (proportional selection Fogel 2000) which favours the fittest training patterns. Thus, CGA progressively focuses the learning process on the most problematic examples, whilst the bulk of the training set is rarely used once it is learned. Using an artificial data set, Paredis (1995) showed that after an initial phase, the examples of highest fitness concentrate near the boundaries between different classes. That is, the candidate solutions learn first a broad partition of the decision space and then refine the boundaries between classes.

By ignoring learned examples, CGA reduces substantially the number of training data evaluations, thus addressing the problem of the computational complexity of evolutionary MLP learning procedures. Paredis used CGA to train MLPs to solve classification (Paredis 1995) and process control (Paredis 1998) problems. In the latter case, a population of MLP controllers was pitched against a population of process starting points in a bioreactor control task. Paredis (1995, 1996) proved the advantages of CGA over the traditional GENITOR genetic algorithm in terms of learning accuracy and speed (Paredis 1995).

To the best of the author's knowledge, Paredis' predator–prey approach for MLP training has not been replicated in the EA literature. A possible explanation may be the difficulty of pairing up two competing evolutionary procedures. Mitchell (2006) and Watson and Pollack (2001) pointed out possible problems of competitive co-evolution procedures, such as loss of gradients, over specialisation, and red queen dynamics. Loss of gradients occurs when either the predators or the prey overpowers the other species, for example, if a subpopulation of training patterns is able to defeat all classifiers. Overspecialisation happens when predators and prey get stuck into some partial solution to the problem. Red queen dynamics develop when predators and prey transition between stages where at turn one species defeats the other, without overall progress towards a more general solution.

Other task-specific problems may affect competitive MLP training approaches like CGA. In particular, the patterns that emerge as the most difficult may be statistical outliers, or the result of some measurement error. The learning process may thus be biased by a few noisy or atypical data samples, and the MLP classifier may fail to generalise. Also, different classes may have overlapping features, that is, they may not be separable. In this latter case, there will always be training patterns able to keep the MLP solutions under attack, and the learning process may fail to converge.

3 Aims and objectives

The aim of this paper is to analyse the performance of the co-evolutionary approach to MLP training in terms of accuracy and computational complexity. Two alternative predator–prey approaches are considered, the first pitching a population of individual solutions against a population of classifiers (Paredis' original approach) and the second opposing a population of training data sets against a population of classifiers (new approach).

The learning accuracy and computational complexity of these two procedures will be compared to those obtained by the standard backpropagation rule (baseline algorithm) and a standard EA. This first set of experiments replicates and expands Paredis' studies on competitive co-evolution, including more data sets and a more precise measure of computational complexity.

In addition, the robustness of the competitive co-evolutionary approach to noisy data sets and overlapping data classes will be evaluated. As discussed in the previous section, noisy or non-separable data patterns amount to hard or impossible to capture prey, which may disrupt or bias the MLP learning process. It is therefore important to evaluate the viability of the co-evolutionary approach beyond Paredis' clear-cut case of disjoint classes (Paredis 1995, 1996). This second set of experiments represents an original contribution of this paper.

The last set of experiments tests the hypothesis that predator–prey algorithms may be particularly suited to deal with unbalanced data classes. The idea is that competitive co-evolution is able to prevent the members of the most numerous class(es) from monopolising the inductive MLP learning procedure. This hypothesis and its experimental verification are original contributions of this paper.

It is important to underline that this paper does not aim to endorse any particular kind of evolutionary algorithm, but to investigate the strengths, shortcomings, and overall viability of competitive co-evolution. For this reason, the two co-evolutionary algorithms under comparison and the standard EA share the same predator evolution procedure. As a consequence, the experimental results should reveal the impact of the two prey evolution approaches tested. Other evolutionary procedures can be used for MLP evolution. It is out of the scope of this paper to investigate the impact of the many instances of evolutionary optimisation methods for ANN learning.

4 Training algorithms

This section introduces the two predator–prey algorithms that will be investigated in this study. The first is the Co-Adaptive artificial Neural Network Training (CANNT) algorithm. It is modelled on CGA and is based on an evolving population of MLP classifiers and a co-adapting set of training data. The second algorithm is the Co-Evolutionary artificial Neural Network Training (CENNT) algorithm. It is based on two co-evolving populations: classifiers and training subsets.

The two algorithms are composed of a module (*Predator*) for the evolution of the MLP classifiers, a module (*Prey*) for the adaptation/evolution of the training patterns, and an interface (*Interaction* module) where the classifiers and training patterns are paired up for evaluation and BP training. The *Predator* and *Prey* modules act in parallel; the *Interaction* module executes after the *Predator* and *Prey* modules. An evolution cycle (generation) includes the execution of all three modules. Both algorithms stop after a pre-fixed number of generations have elapsed, and the best performing classifier is taken as the final solution.

The two competitive algorithms share the same *Predator* module and have their own specific *Prey* and *Interaction* modules. Their flow charts are shown in Fig. 1.

4.1 Co-Adaptive ANN Training algorithm (CANNT)

The CANNT algorithm uses the same co-evolutionary scheme employed in CGA, featuring a population of MLP classifiers (the predators) competing against the individual training patterns (the preys). The main differences between CGA and CANNT are the kind of evolutionary procedure used and the fitness evaluation procedure.

The *Predator* module of the CANNT algorithm is an evolutionary algorithm for MLP training. The *Prey* module of the CANNT algorithm contains the population of training data. The training data are not a truly evolving population, since only the individual fitness values are updated.

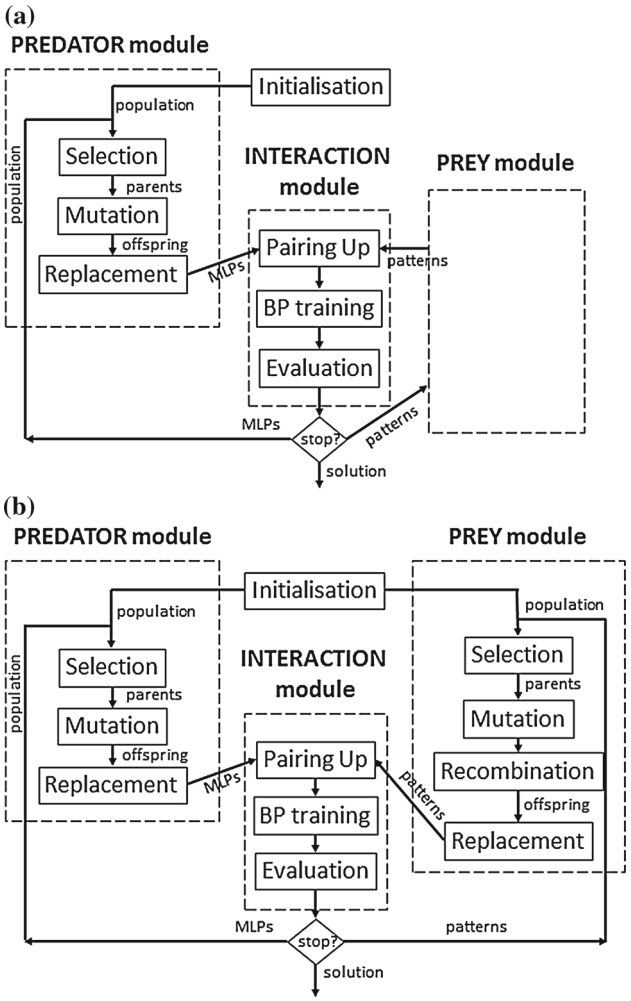

Fig. 1 Flow charts of competitive MLP training algorithms. **a** Flow chart of CANNT algorithm. **b** Flow chart of CENNT algorithm

4.1.1 CANNT predator module

In CGA, the MLP solutions are evolved using a genetic algorithm based on floating point representation, steady-state population replacement (Fogel 2000), proportional selection (Fogel 2000), two-point crossover, and an adaptive mutation operator (Whitley 1989).

Also in CANNT, the population is encoded using floating point representation. Generational replacement (Fogel 2000) is adopted in CANNT for the population of ANN classifiers, in order to foster exploration of the solution space. That is, instead of replacing at each generation the least fit individual with a new individual of higher fitness (steady-state replacement), the whole parent population is replaced with an equal number of offspring (generational replacement). Only the fittest solution survives into the next generation (elitism Fogel 2000).

The selection scheme used in CANNT is fitness ranking (Fogel 2000), which eliminates the fitness scaling problem,

and avoids premature convergence on suboptimal solutions (Pham and Castellani 2010).

Genetic mutations in CANNT modify the weights of all the nodes of a candidate MLP solution. For each weight, the perturbation is randomly sampled with uniform probability from a small interval $[-\delta, \delta]$ of fixed width 2δ. To avoid the competing conventions problem (Thierens et al. 1993), genetic recombination is not featured in the evolution of the ANN classifiers. Given the lack of genetic crossover and the real-valued encoding of the solutions, the MLP training procedure is akin to Evolutionary Programming (Fogel 2000).

At the beginning of the search, the initialisation procedure generates the starting population of MLP solutions. The structure of the ANN is pre-set by the user, whilst the weights are randomly initialised.

4.1.2 CANNT prey module

The *Prey* module contains the set of training patterns. The patterns do not evolve, their only feature is the individual fitness which is calculated in the *Interaction* module. At initialisation, all M training patterns $v_j (j \in [1, M])$ are assigned fitness $f\left(v_j, 0\right) = 0.5$.

4.1.3 CANNT interaction module

The new generation of predators (MLPs) is paired up with a sample of prey (patterns) for training and evaluation.

For each of the N candidate MLP solutions p_i $(i \in [1, N])$, the training set T is scanned. For each of the M elements of the training set v_j $(j \in [1, M])$, a random number $rnd \in [0, 1]$ is drawn and v_j is picked if $rnd \leq f\left(v_j\right)$, where $f\left(v_j\right) \in [0, 1]$ is the fitness of the jth pattern v_j. That is, for each predator the prey are selected with a probability that is proportional to their fitness. At the end of the pairing process, each MLP p_i is paired with a different subset $K_i \subseteq T$ of randomly picked training patterns. Different training subsets may have a different number of data samples.

Each subset K_i is first used to train the associated p_i, in order to speed up the learning process (Lamarckian learning). Since BP learning is computationally expensive, only one cycle of the procedure is executed for each solution. Because the probability of selection for the training data is proportional to fitness, the Lamarckian learning process focuses primarily on the most difficult examples.

Since the MLP population is almost completely renewed at every generation, lifetime fitness evaluation (Paredis 1996) cannot be used. In CANNT, the fitness of the predators and prey is evaluated at once in the *Interaction* module. The fitness $f\left(p_i\right)$ of a candidate MLP solution p_i paired up to subset K_i is equal to its classification accuracy:

$$f\left(p_i\right) = \frac{c_i\left(K_i\right)}{S_i} \tag{1}$$

where $c_i\left(K_i\right)$ is the number of patterns in K_i correctly classified by p_i, and S_i is the size of the training subset K_i. A training pattern v_j can be included in several training subsets. The fitness $f\left(v_j, t+1\right)$ of training pattern v_j at iteration t+1 depends on its previous fitness $f\left(v_j, t\right)$, and on the frequency v_j is not classified correctly by the MLP solutions in the current iteration:

$$f\left(v_j, t+1\right) = 0.8 \cdot f\left(v_j, t\right) + 0.2 \cdot \frac{m_j(t)}{P_j(t)} \tag{2}$$

where $m_j(t)$ is the number of times v_j was misclassified at iteration t, and $P_j(t)$ is the number of classification attempts (the number of training subsets v_j belongs to). That is, the fitness of a predator corresponds to how many preys it captures out of a sample of the prey population; and the fitness of a prey depends on how many times it escapes predation out of the total number of predation attempts. Note that at any time t, fitness $f\left(v_j, t\right) \in [0, 1]$.

The first term at the right hand side of Eq. (2) represents the trace of the past evaluations of the pattern. It simulates Paredis' lifetime fitness evaluation mechanism. It was found experimentally that the trace helps to prevent excessive oscillations in the co-evolution process. The 8:2 ratio between the contributions of the trace and the last evaluation was set experimentally. This ratio can be changed to increase or decrease the speed at which the system forgets the result of past fitness estimates.

4.2 Co-evolutionary ANN training algorithm (CENNT)

The CENNT algorithm features two co-evolving populations: MLP solutions and training subsets of examples.

4.2.1 CENNT predator module

CENNT uses the same *Predator* module used by CANNT and described in Sect. 4.1.1.

4.2.2 CENNT prey module

The *Prey* module of the co-evolutionary CENNT algorithm is an evolutionary algorithm for generation of training pattern subsets. The aim of the *Prey* module is to evolve subsets K_i of training patterns that are difficult to classify.

Each solution v_j is encoded as a bit string of length equal to the number M of training patterns. If the jth element (gene) of the string (chromosome) is set to '1' (allele), the corre-

sponding training pattern v_j is included in the training subset, otherwise ('0' allele) is excluded.

The module employs generational replacement and the fitness ranking selection routine. New individuals are generated through the standard bit flip mutation and two-point crossover (Fogel 2000) operators. The mutation probability depends on a predefined mutation rate m_v. If an offspring is selected for mutation, each of its genes is mutated with a predefined probability m_g.

If an offspring corresponding to the empty set is generated, it is re-initialised with equal allele probability per each gene.

At initialisation, the solutions are randomly initialised with equal allele probability per each gene.

Overall, the *Prey* module of CENNT is akin to a standard genetic algorithm (Fogel 2000).

4.2.3 CENNT interaction module

Each offspring of the predator population (MLPs) is paired up with one or more of the offspring of the prey population (training subsets).

For each of the N predators (candidate MLP solutions) p_i ($i \in [1, N]$), one prey v_j (training subset K_j) is randomly picked. All predators are paired up to at least one prey, and vice versa. If the number of predators is equal to the number of prey, each predator is paired up to one and just one prey. If the predators (prey) are less numerous than the prey (predators), a predator (prey) can be paired up to more than one prey (predator). To support the uniformity of the evaluation procedure, no predator (prey) can be paired up to $n + 1$ prey (predators) until all the other predators (prey) are paired up to at least n prey (predators).

Once ANNs and training subsets are associated, one cycle of ANN Lamarckian BP learning (Sect. 4.1.3) is executed per each associated subset. Because the prey evolution process favours the creation of subsets of difficult examples, Lamarckian learning focuses primarily on the most difficult examples.

The fitness $f(p_i)$ of a predator p_i is equal to its classification accuracy:

$$f(p_i) = \frac{1}{n} \cdot \sum_{j=1}^{n} \frac{c_i\left(K_j\right)}{S_j} \tag{3}$$

where $c_i\left(K_j\right)$ is the number of patterns in K_j correctly classified by the candidate MLP solution p_i, S_j is the size of K_j (i.e. the number of training patterns included in K_j), and n is the number of prey paired up to p_i. The fitness of a prey v_j is equal to:

$$f\left(v_j\right) = 1 - \frac{1}{m} \cdot \sum_{i=1}^{m} \frac{c_i\left(K_j\right)}{S_j} \tag{4}$$

where m is the number of predators paired up to v_j. If $m = n$, then $f(p_i) = 1 - f\left(v_j\right)$ (complementary fitness).

5 Methods

The main goal of this paper is to ascertain the viability and effectiveness of the competitive approach through a series of experimental tests. The aim of each test is to train an MLP to solve a given classification problem. In all cases, the ANN is composed of one hidden layer of neurons.

In all tests, the performance of the CANNT and CENNT algorithms is compared with that of the standard BP rule and a conventional EA algorithm. The conventional EA corresponds to the *Predator* module of CANNT and CENNT. The only difference between this EA and the two competitive algorithms is that the solutions are trained and evaluated (see *Interaction* modules of CANNT and CENNT) using the whole training set of examples. The conventional EA is henceforth referred to as the 'standard' or 'traditional' EA.

The first set of experiments evaluates the effectiveness and efficacy of the co-evolutionary approach on seven classification benchmarks. The first classification benchmark is Paredis' artificial classification problem (Paredis 1995, 1996). In its original formulation, Paredis' test problem is composed of 200 artificial data divided into four classes defined in the R^2 domain $D = [-1, 1] \times [-1, 1]$. They are partitioned as follows:

$$\text{class} = \begin{cases} 1 & \text{if } (x, y) \in D \& x^2 + y^2 < 0.25 \\ 2 & \text{if } (x, y) \in D \& (y > 1 - x \| y < -1 - x) \\ 3 & \text{if } (x, y) \in D \& (x > 0.5 \& y < 0 \| x > 0 \& y < -0.5) \\ 4 & \text{else} \end{cases} \tag{5}$$

Since Paredis' original set is not available, it was simulated randomly generating 500 artificial data patterns in the domain D, and labelling these data according to Eq. (5). This data set is henceforth referred as $N00$ and is visualised in Fig. 2a. The $N00$ set is split into a training set containing 80% of the data patterns (400 elements), and a test set containing the remaining 20% of the data patterns (100 elements).

In addition to Paredis' artificial problem, the four test algorithms (CANNT, CENNT, standard EA, and BP) were evaluated on six well-known real-world classification benchmarks taken from the UCI Machine Learning Repository (Lichman 2013), namely *Ionosphere*, *Iris*, *Musk "Clean2"* (henceforth *Musk*), *Pen-Based Recognition of Handwritten Digits* (henceforth *Pen*), *Image Segmentation* (henceforth Segmentation), and *Vehicle Silhouettes* (henceforth *Vehicles*).

The four data classes featuring in the $N00$ data set are disjoint. Unfortunately, many real-world problems are char-

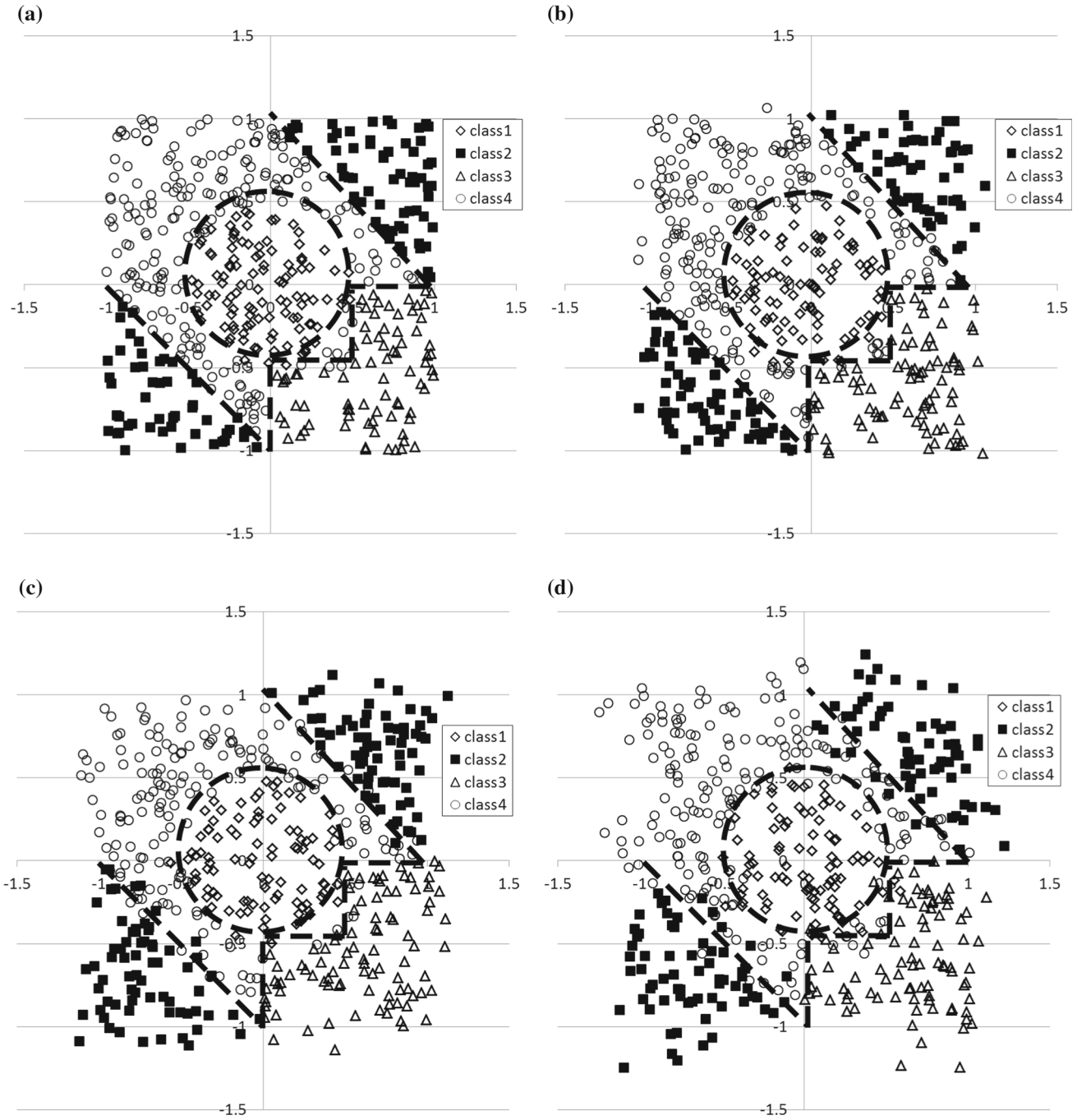

Fig. 2 Artificial data sets: separable and overlapping classes. **a** *N*00, no noise (Paredis 1995, 1996). **b** *N*10, 10% noise. **c** *N*20, 20% noise. **d** *N*30, 30% noise

acterised by overlapping classes; a typical example being the *Iris* data set where two of the three flower types have overlapping features. As mentioned in Sects. 2 and 3, non-disjoint data classes may be problematic for competitive approaches, since 100% accuracy is not attainable, and there will always be prey able to evade the predators. It is therefore important to investigate the ability of co-evolutionary algorithms to deal with such data sets. Although some of the real-world benchmarks employed in this study are known for including non-separable classes, the degree and nature of the overlapping is not easily quantifiable.

In the second set of experiments, the four test algorithms are evaluated on artificial data sets characterised by controlled levels of class overlapping. Two cases are taken in consideration: the data categories are intrinsically overlapping (e.g. Iris data set), or the data categories are disjoint and

Table 1 Data sets

	Total examples	Training examples	Input features	Output classes
N00-30, NT10-30	500	400	2	4
Iris	150	80%	4	3
Ionosphere	351	200	33	2
Musk	6598	80%	166	2
Pen	10, 992	7494	16	10
Segment	2310	80%	19	7
Vehicle	846	80%	18	4

Table 2 MLP structures

Data set	Input-hidden-output nodes	Transfer function-hidden nodes	Transfer function—output nodes
N00-30, NT10-30	2–30–4	Hyper-tangent	Sigmoidal
Iris	4–2–3	Hyper-tangent	Sigmoidal
Ionosphere	33–2–2	Hyper-tangent	Sigmoidal
Musk	166–20–2	Hyper-tangent	Sigmoidal
Pen	16–60–10	Hyper-tangent	Sigmoidal
Segment	19–30–7	Hyper-tangent	Sigmoidal
Vehicle	18–30–4	Hyper-tangent	Sigmoidal

the overlapping is due to random noise in the data acquisition process. In the former case, some degree of classification inaccuracy is unavoidable. In the latter, it may be still possible to train the MLP to learn the correct partition of the input space, and achieve 100% recall accuracy on ideally uncorrupted data. In this latter case, inaccuracy may occur if the classifier learns to reproduce the noise in the training patterns (data *overfitting*).

To represent intrinsically overlapping categories (first case), three data sets were created using the partition described in Eq. (5), and perturbing the (x, y) coordinates of each data sample by some random noise

$$z' + z + \delta \tag{6}$$

where $z = x, y \in [-1, 1]$, and $\delta \in [-\sigma, \sigma]$ is a random number sampled with uniform probability from the interval $[-\sigma, \sigma]$. In the first case ($N10$ data set) $\sigma = 0.1$ (10% noise), in the second ($N20$ data set) $\sigma = 0.2$ (20% noise), and in the third ($N30$ data set) $\sigma = 0.3$ (30% noise). The $N10$, $N20$, $N30$ sets represent partitions of increasing degree of overlapping between classes, and are shown in Fig. 2b–d, respectively. The data sets are composed of 400 training patterns and 100 test patterns (500 total data).

To represent the noisy samples of intrinsically disjoint classes (second case), three data sets were created using the partition described in Eq. (5). The data sets were then split into a training set containing 80% of the data patterns (400 elements), and a test set containing the remaining 20% of the data patterns (100 elements). Only the coordinates of the data patterns belonging to the training set were perturbed as in Eq. (6), adding to the position coordinates of each training sample, respectively, 10% ($\sigma = 0.1$, *NT10* data set), 20% ($\sigma = 0.2$, *NT20* data set), and 30% ($\sigma = 0.3$, *NT30* data set) noise. The 100 patterns belonging to the test set were not perturbed.

For consistency of training, the input data of all sets (artificial and real-world benchmarks) are pre-processed using the mean-variance procedure. In some cases (artificial sets, *Ionosphere*, *Pen*) the data sets come pre-divided into training and test set. In the remaining cases, before the algorithm is run 80% of the data are randomly picked for training and the remaining 20% are used for validation. The main features of the data sets are given in Table 1. The population size of the *Predator* module (CANNT, CENNT, and EA) is always fixed to 100 individuals. The other parameters of the CANNT and CENNT algorithms and the MLP structure were optimised experimentally. They are given in Tables 2 and 3.

6 Experimental results

On each data set, each algorithm was run 10 times and the descriptive statistics (minimum, maximum, median, and 10th and 90th percentiles) of the results are reported in Table 4. The significance of the differences between the results obtained by the four algorithms was pairwise analysed using Mann–Whitney U tests. For each benchmark, the best learning result (highest MLP classification accuracy) is highlighted in bold, and the table columns of the results that are not significantly different from the best are shaded in grey.

Table 3 BP and EA parameters

Learning algorithm settings	BP	Predator module	Prey module	Interaction module
(a) Main parameters				
Learning coefficient	0.1	–	–	0.1
Momentum term	0.01	–	–	0.01
Initial range for MLP weights	[−0.05, 0.05]	[−0.05, 0.05]	–	–
MLP weights mutation rate	–	0.05	–	–
Amplitude MLP weights mutation	–	0.1	–	–
Backpropagation operator rate	–	–	–	1.0
Backpropagation operator cycles	–	–	–	1
Crossover rate	–	–	1.0	–
Prey mutation rate (m_v)	–	–	0.05	–
Gene mutation rate m_g	–	–	0.1	–

'–' denotes not applicable

Data set	Cycles
(b) Number of learning cycles	
N00-30, NT10-30	10,000
Iris	2000
Ionosphere	2000
Musk	2000
Pen	7000
Segmentation	7000
Vehicle	7000

Data set	Predator module	Prey module
(c) CENNT population sizes		
N00	100	200
N10	100	100
N20	100	50
N30	100	100
NT10	100	100
NT20	100	200
NT30	100	200
Iris	100	100
Ionosphere	100	50
Musk	100	200
Pen	100	50
Segmentation	100	200
Vehicle	100	50

Table 4 Classification benchmarks—accuracy results

ARTIFICIAL SET *N00*				
	BP	EA	CENNT	CANNT
min	92.08	94.06	94.06	91.09
10th percentile	92.97	94.95	94.06	93.76
median	95.05	**96.53**	95.05	95.54
90th percentile	97.03	97.13	99.11	99.01
max	97.03	98.02	100.00	99.01
IONOSPHERE				
min	94.70	94.70	88.74	92.05
10th percentile	95.30	94.70	91.72	92.05
median	**95.36**	**95.36**	93.38	95.03
90th percentile	96.09	96.09	94.90	96.09
max	96.69	96.69	96.69	96.69
IRIS				
min	93.33	83.33	90.00	90.00
10th percentile	93.33	89.33	93.00	90.00
median	**96.67**	93.33	**96.67**	93.33
90th percentile	100.00	97.00	100.00	100.00
max	100.00	100.00	100.00	100.00
MUSK				
min	98.86	98.71	98.56	98.64
10th percentile	99.00	98.85	98.76	98.84
median	**99.43**	99.17	99.17	99.13
90th percentile	99.71	99.56	99.28	99.63
max	99.85	99.70	99.62	99.70
PEN				
min	92.54	92.54	92.60	92.31
10th percentile	92.54	92.56	92.62	92.31
median	92.81	92.78	**92.84**	92.61
90th percentile	92.93	92.97	93.00	97.13
max	93.11	92.97	93.05	97.54
SEGMENTATION				
min	96.10	94.37	96.32	96.10
10th percentile	96.49	96.13	96.71	96.30
median	96.97	97.08	97.51	**97.73**
90th percentile	97.84	97.62	98.05	98.29
max	97.84	97.62	98.05	98.48
VEHICLE				
min	79.29	79.88	78.11	79.88
10th percentile	79.29	80.95	79.70	80.95
median	83.14	83.14	82.25	**83.43**
90th percentile	85.21	85.80	84.08	85.38
max	85.21	85.80	84.62	86.98

6.1 Classification accuracy

On the seven data sets ($N00$ and UCI benchmarks), there are no significant differences in the average accuracy results obtained by the four algorithms. Figure 3 compares the learning curves of the three evolutionary algorithms (EA, CENNT, and CANNT) on the $N00$ benchmark problem. Also in this case, the three algorithms appear to perform comparably and the three plots show similar population histories.

Overall, the accuracy results one the UCI data sets are comparable with the state-of-the-art in the literature (Castellani 2013). Likewise, the accuracy obtained by the four algorithms on the artificial $N00$ data set is in good agreement with the 96.45% accuracy obtained by CGA (Paredis 1996).

6.2 Computational complexity

The most costly computational procedures in the evolutionary training of the MLP solutions are the forward processing of the training patterns through the ANN structures (fitness evaluation and Lamarckian learning), and the backward prop-

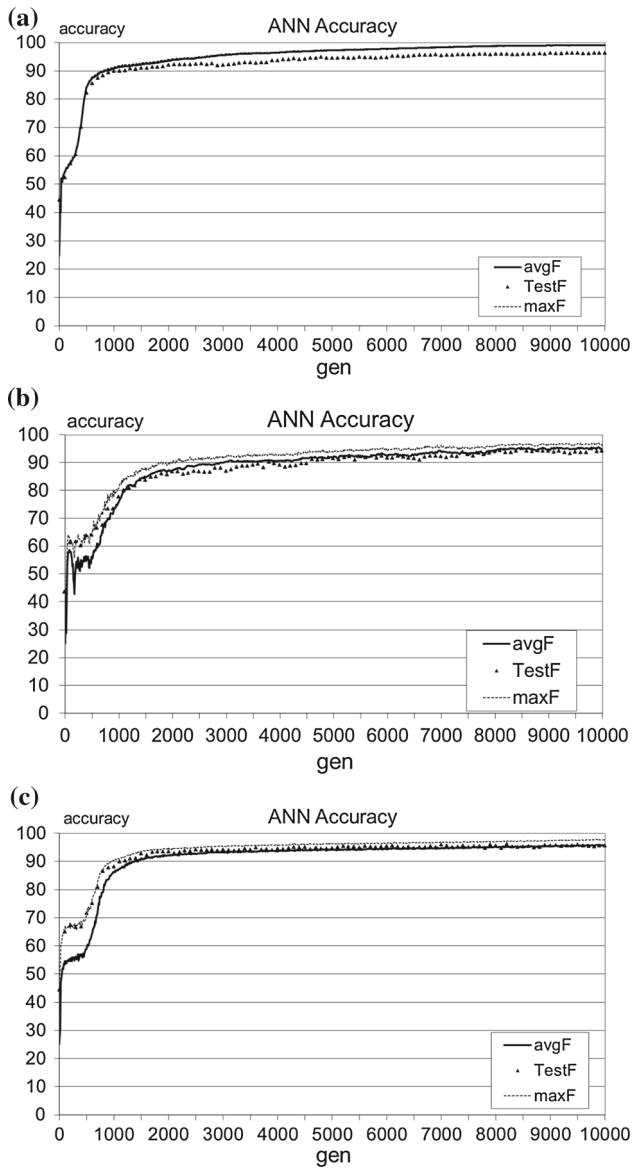

Fig. 3 Artificial data set, $N00$ (no noise)—learning curves. *Each plot* reports the average (avgF) classification accuracy of the population on the training set and the accuracy of the best individual on the training (maxF) and test set (TestF). **a** EA, **b** CENNT, **c** CANNT

agation of the output errors (Lamarckian learning). Hereafter, the cost for passing once (forward or backward) one input pattern through the MLP will be taken as the unit of computational cost. Table 5a reports for 10 runs of each algorithm, the statistical median of the total number of training data passes per run. The entry of the least computationally expensive evolutionary procedure is highlighted in bold.

For ease of reference, the cost (number of passes) has been normalised with respect to the cost of BP in Table 5b. That is, for each algorithm, the value reported in each cell of the table corresponds to the number of data passes performed by that algorithm for one pass performed by the BP algorithm.

Table 5 Computational and time costs

	BP	EA	CENNT	CANNT
(a) Sum of forward and backward passes of the training set				
Evaluations ($\times 10^6$)				
Noise = 0% ($N00$)	8	1200	405.76	**155.25**
Ionosphere	0.80	120	19.05	**14.04**
IRIS	0.48	72	6.41	**3.74**
Musk	21.11	3167	1833.91	**158.81**
Pen	104.92	15737	6770.58	**1088.86**
Segmentation	25.87	3881	1910.95	**258.88**
Vehicle	9.48	1422	326.34	**177.76**
(b) Sum of forward and backward passes of the training set. The values have been normalised with respect to BP				
Evaluations				
NOISE = 0% ($N00$)	1	150	50.72	**19.41**
Ionosphere	1	150	23.81	**17.55**
IRIS	1	150	13.36	**7.79**
Musk	1	150	86.87	**7.52**
Pen	1	150	64.53	**10.38**
Segmentation	1	150	73.86	**10.01**
Vehicle	1	150	34.43	**18.75**
(c) Noise = 0% ($N00$) data set, running time				
Algorithm running time				
Time (s)	18	2518	657	432
Normalised	1.00	139.86	36.47	23.97
(d) Noise = 0% ($N00$) data set, ratio between normalised number of evaluation and time				
Evaluations versus time				
Evaluations	1	150.00	50.72	19.41
Time	1	139.86	36.47	23.97
Evaluations/time	1	1.07	1.39	0.81

The entry of the least computationally expensive evolutionary procedure is highlighted in bold.

Table 5c gives an example of the running times of the four algorithms on the $N00$ benchmark. The example refers to the average running time of each procedure, calculated as the statistical median of 10 independent runs. For ease of reference, the running times are also normalised to the running time of the BP algorithm.

Finally, Table 5d compares the total forward and backward passes (Table 5b, normalised figure) and average running time (Table 5c, normalised figure) totalled by the four algorithms on $N00$ data set. The figures show that the sum of forward and backward passes is a good proxy for the computational complexity and hence running time of the algorithms.

Figure 4 shows the evolution of the training subset size for the CENNT and CANNT algorithms on the $N00$ benchmark problem. The plots show that the adaptive CANNT procedure is much faster than the co-evolutionary CENNT

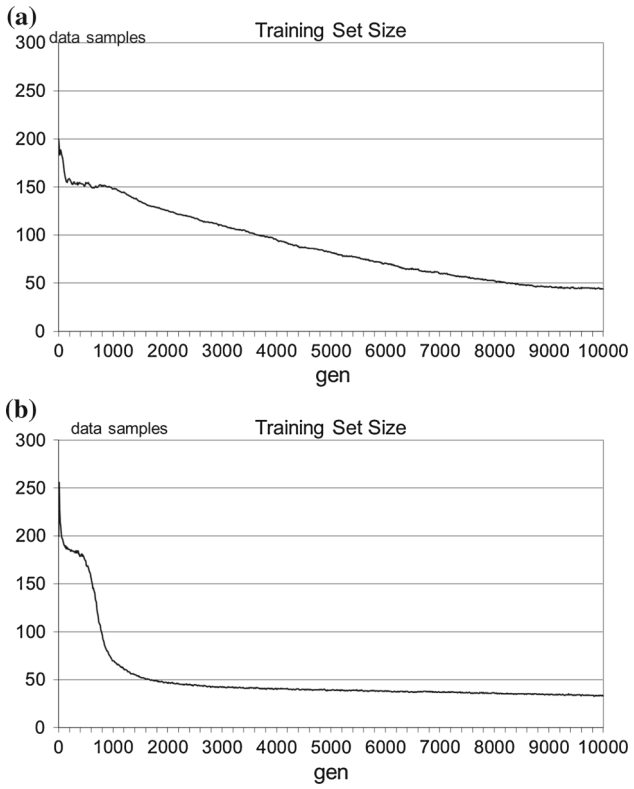

Fig. 4 Artificial data set, *N*00 (no noise)—evolution of training subset size. **a** CENNT. **b** CANNT

in reducing the training data set size. After less than 2000 generations, CANNT has already restricted the training effort on a number of patterns that is one-tenth of the size of the full training set. In contrast, it takes nearly the whole training procedure to CENNT to achieve a similarly small training subset size. Manual inspection confirmed Paredis's observation that the selected training patterns tend to lie at the boundaries between classes (Paredis 1995).

Figure 5 visualises the trade-off between learning accuracy and computational costs of the four algorithms on the N00 and UCI data sets. Overall, Table 5a–c and Figs. 4 and 5 demonstrate the ability of CANNT and to a lesser extent of CENNT to focus quickly on the critical data samples. This ability is reflected in considerable savings in computation time with respect to the EA using the full training set.

6.3 Intrinsically overlapping data classes

Table 6 shows the results of the learning trials on intrinsically non-disjoint (training and test set) data classes with the same conventions as in Table 4. For ease of reference, the results on the *N*00 data set have been duplicated. Also in this case, the four algorithms give statistically undistinguishable training results. As the data classes increasingly overlap, the performance of the algorithms drops. Figure 6a plots the accuracy of the four algorithms normalised to the accuracy obtained on the uncorrupted *N*00 data set.

6.4 Noisy data, overfitting test

Table 7 shows the results of the learning trials on noisy training data (uncorrupted test set) with the same conventions as in Table 4. For ease of reference, the results on the *N*00 data set have been duplicated. Figure 6b shows the deterioration of the MLP learning accuracy as the noise in the training set is increased. Compared to the case of intrinsically non-disjoint data categories, the case of noisy training patterns shows a more graceful decrease of the learning accuracy. This result highlights the generalisation capability of this kind of classifier.

The two co-evolutionary learning procedures (CANNT and CENNT) appear the least affected by noise on the *NT10*-30 benchmarks (Fig. 6b). The BP algorithm fails to perform optimally on the noisy *NT10*-30 sets, obtaining statistically significantly worst learning accuracies than the three evolutionary procedures. This result confirms the superior robustness to noise of global optimisation procedures compared to greedy gradient-based search, indicating that BP is more liable to overfitting than the evolutionary methods.

7 Unbalanced data sets

A common problem in many classification tasks is that the number of examples per data category is not uniform. That is, some classes are over-represented with respect to others in the training set. In this case, the inductive learning process is monopolised by the largest classes, whilst the smallest ones tend to be overlooked (Chawla et al. 2004). This bias in the learning procedure leads to suboptimal classification performance.

Class unbalance affects the learning process as well as the evaluation of the classifier, since the overall accuracy measure is mostly determined by the most numerous categories. In the specific case of EAs, class unbalance affects the effectiveness of the fitness evaluation procedure and Lamarckian learning operator.

In his overview on unbalanced data mining, Weiss (2004) reviews the most common remedies to the problem, such as under-sampling (representatives of the most numerous classes are removed), oversampling (representatives of the least numerous classes are artificially created, e.g. samples are duplicated), cost-sensitive learning (the cost of false positives and negatives depends on the class size), and boosting (examples difficult to classify are weighted more than easy ones). These methods alter the data distribution either directly by adding or removing examples, or indirectly by weighting differently the cost of errors. They generally

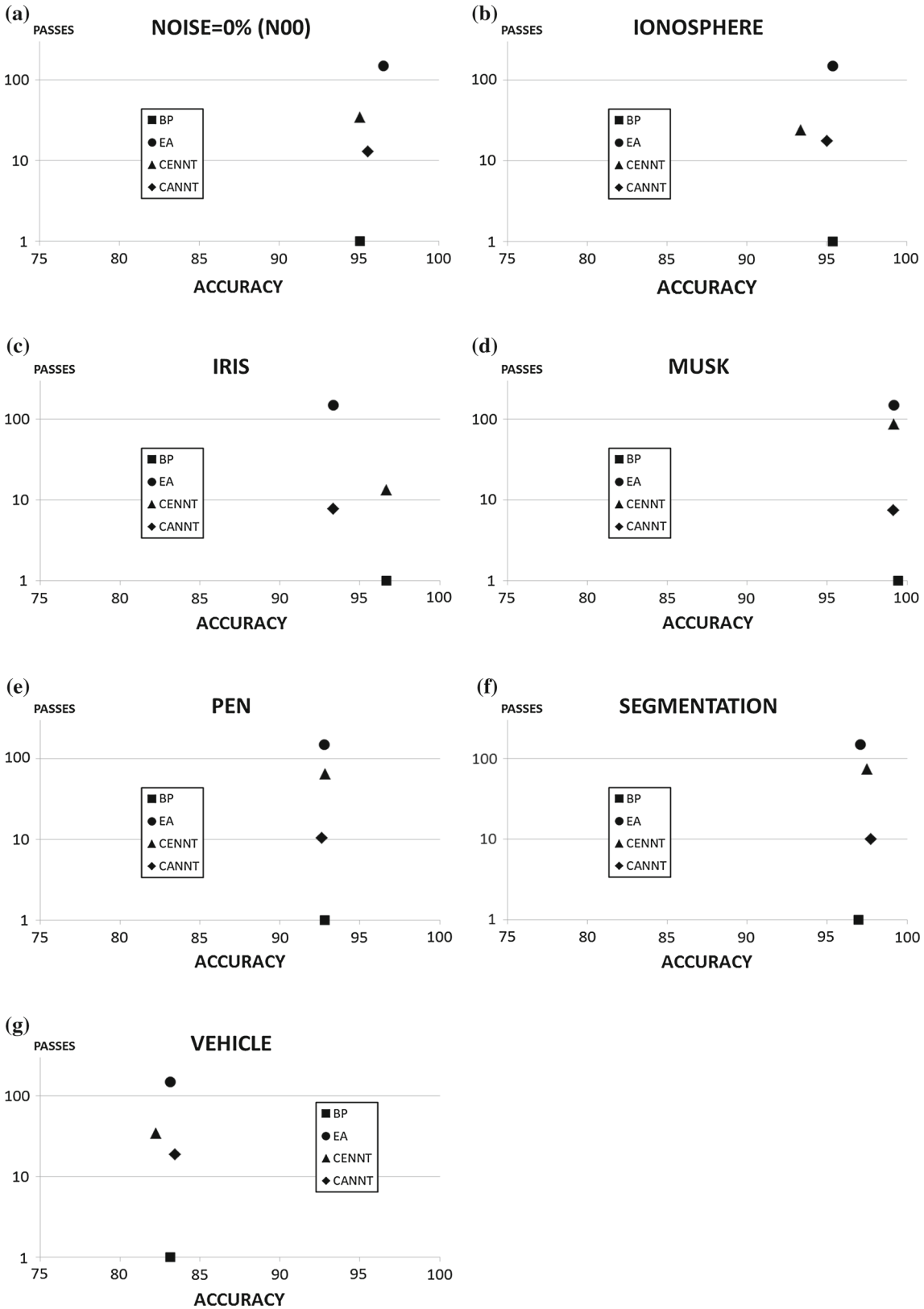

Fig. 5 Classification benchmarks: accuracy versus number of forward and backward passes (normalised to BP). The number of passes is shown using a logarithmic scale. **a** N00 data set. **b** Ionosphere data set. **c** Iris data set. **d** Musk data set. **e** Pen data set. **f** Image segmentation data set. **g** Vehicle data set

Table 6 Intrinsically overlapping data classes (training and test set)—accuracy results

	BP	EA	CENNT	CANNT
ARTIFICIAL SET *N00* (NOISE=0%)				
min	92.08	94.06	94.06	91.09
10th percentile	92.97	94.95	94.06	93.76
median	95.05	**96.53**	95.05	95.54
90th percentile	97.03	97.13	99.11	99.01
max	97.03	98.02	100.00	99.01
ARTIFICIAL SET *N10* (NOISE=10%)				
min	91.00	92.00	91.00	88.00
10th percentile	91.90	92.00	91.90	90.70
median	93.00	**95.50**	93.50	93.00
90th percentile	95.30	99.00	96.00	96.00
max	98.00	99.00	96.00	96.00
ARTIFICIAL SET *N20* (NOISE=20%)				
min	85.00	83.00	84.00	87.00
10th percentile	85.90	86.60	84.90	87.90
median	89.00	90.00	88.50	**90.50**
90th percentile	91.00	92.20	91.20	93.30
max	91.00	94.00	93.00	96.00
ARTIFICIAL SET *N30* (NOISE=30%)				
min	84.00	85.00	83.00	82.00
10th percentile	84.90	85.90	83.00	84.70
median	86.00	**88.00**	87.00	87.00
90th percentile	90.10	90.00	90.10	88.10
max	91.00	90.00	91.00	89.00

require ad-hoc decisions on how to readdress the data unbalance (cost-sensitive and re-sampling methods), and may lead to overfitting (all), large training sets (oversampling), or loss of important data (under-sampling).

In this study, competitive co-evolution is evaluated as a possible alternative method to deal with unbalanced data classes. Similarly to boosting, co-evolutionary algorithms adjust adaptively the fitness of the most difficult data samples (CANNT, CGA), or whole data subsets (CENNT). The fittest elements are used more frequently by the Lamarckian training and fitness evaluation procedures, and thus their contribution to the evolutionary learning process is magnified. At the same time co-evolutionary methods perform a sort of adaptive under-sampling, avoiding repeated use of the easiest samples. Compared to classical under-sampling techniques, competitive EAs do not risk losing important data, since only learned examples are disregarded, and no example is lost forever.

A new data set (*UB*) composed of 1000 data samples was created. The samples were labelled according to the partition of Eq. (5), and divided into a training set of 800 elements and a test set of 200. The training set comprises 250 examples of classes 1, 2, and 4, and 50 examples of class 3. The test set contains 50 examples per class. Figure 7 shows the two data sets.

Ten independent learning trials were performed on the *UB* data set for each of the four algorithms (BP, EA, CENNT, CANNT). The descriptive statistics of the results are calculated, and the significance of the differences is pairwise evaluated using Mann–Whitney *U* tests. The results are reported in Table 8a using the same conventions as in Table 4. Table 8b presents a breakdown of the statistical median of

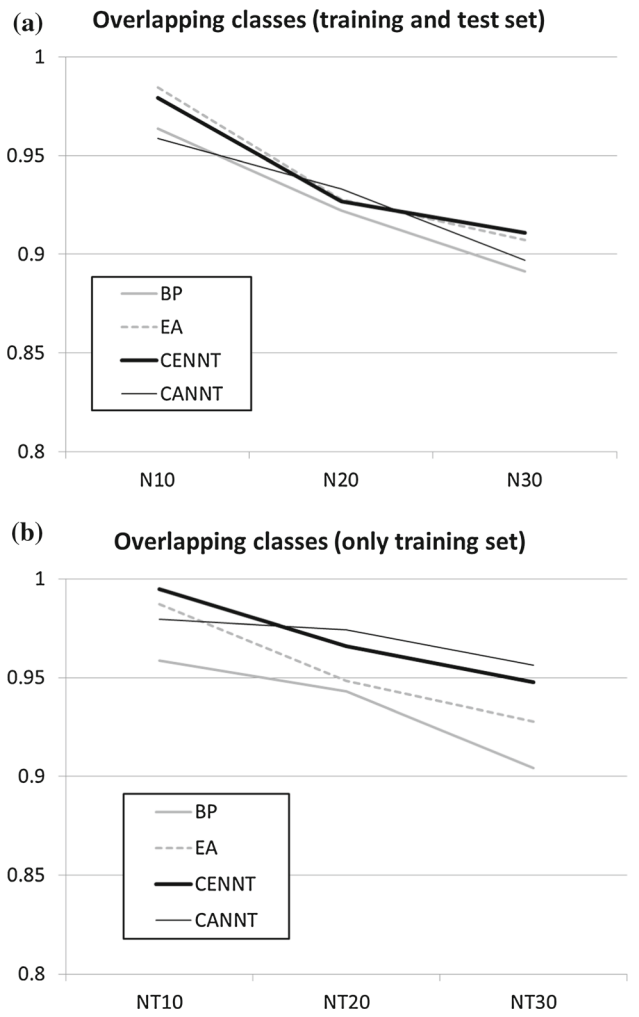

Fig. 6 Deterioration of accuracy with increase in noise. The accuracy obtained on the uncorrupted data sets is normalised to 1. The accuracy obtained on the noisy sets is expressed as a fraction of the accuracy achieved on the uncorrupted sets. **a** *N00-30* data sets. **b** *NT00-30* data sets

the classification accuracy, and visualises the results of the significance tests. Table 9 shows the confusion matrix (statistical median of the classification results) for each algorithm. The latter two tables detail the class distribution and nature of the classification errors.

Table 8a reveals the inferior performance of BP compared to the three EA procedures. This result confirms the superiority of global EA methods over greedy BP search in presence of unbalanced class distribution (Weiss 2004). Tables 8b and 9 show that, on the smallest class 3, the traditional EA does not perform better than BP. That is, the EA learns to recognise the elements of the largest classes marginally better than BP, but does not improve the classification accuracy of BP on the smallest class. The superior learning accuracy of the co-evolutionary methods on the smallest class is apparent in both Tables 8b and 9, and is confirmed by the signifi-

Table 7 Overlapping data classes (only training set), overfitting test—accuracy results

	BP	EA	CENNT	CANNT
ARTIFICIAL SET *N00* (NOISE=0%)				
min	92.08	94.06	94.06	91.09
10th percentile	92.97	94.95	94.06	93.76
median	95.05	**96.53**	95.05	95.54
90th percentile	97.03	97.13	99.11	99.01
max	97.03	98.02	100.00	99.01
ARTIFICIAL SET *NT10* (NOISE=10%)				
min	88.00	92.50	90.00	93.50
10th percentile	90.70	93.40	92.25	93.50
median	92.50	**95.75**	95.00	95.00
90th percentile	94.15	96.05	96.60	96.05
max	95.50	96.50	97.50	96.50
ARTIFICIAL SET *NT20* (NOISE=20%)				
min	84.50	90.50	85.50	91.50
10th percentile	88.55	90.50	89.10	91.50
median	91.00	92.00	92.25	**94.50**
90th percentile	92.05	93.50	95.55	96.05
max	92.50	93.50	96.00	96.50
ARTIFICIAL SET *NT30* (NOISE=30%)				
min	82.50	88.50	85.50	88.50
10th percentile	83.40	88.95	87.30	88.50
median	87.25	90.00	90.50	**92.75**
90th percentile	89.50	92.05	93.10	94.10
max	89.50	92.50	94.00	95.00

cance tests. The results indicate that competitive evolutionary learning is a promising alternative to tackle the problem of unbalanced data distributions.

The best classification accuracy is obtained by the CANNT algorithm. Figure 8 shows a breakdown of the evolution of the learning accuracy for the standard EA and CANNT. The plots show that, compared to the standard EA, CANNT obtains a more steady albeit irregular improvement of the accuracy on the under-represented class 3.

8 Discussion

This study was inspired by Paredis' work on co-evolutionary MLP training, and made a number of contributions to the understanding of this approach. Whilst Paredis applied CGA to a number of test cases (Paredis 1995, 1998, 1996), he never evaluated systematically the capabilities of competitive co-evolution. In this study two alternative procedures were tested: CANNT based on Paredis' approach, and the new CENNT procedure. The performance of the two co-evolutionary algorithms was evaluated in terms of accuracy, computational complexity, ability to deal with non-separable data classes, and robustness to noise. A standard EA and the customary BP rule were used as terms of comparison. This systematic study of co-evolutionary MLP training constitutes the first contribution made in this paper.

Overall, the learning results of the four procedures are comparable with the state-of-the-art in the literature. The BP rule is clearly the least computationally intensive procedure. However, the two co-evolutionary algorithms showed superior robustness to data overfitting. This superiority was shared

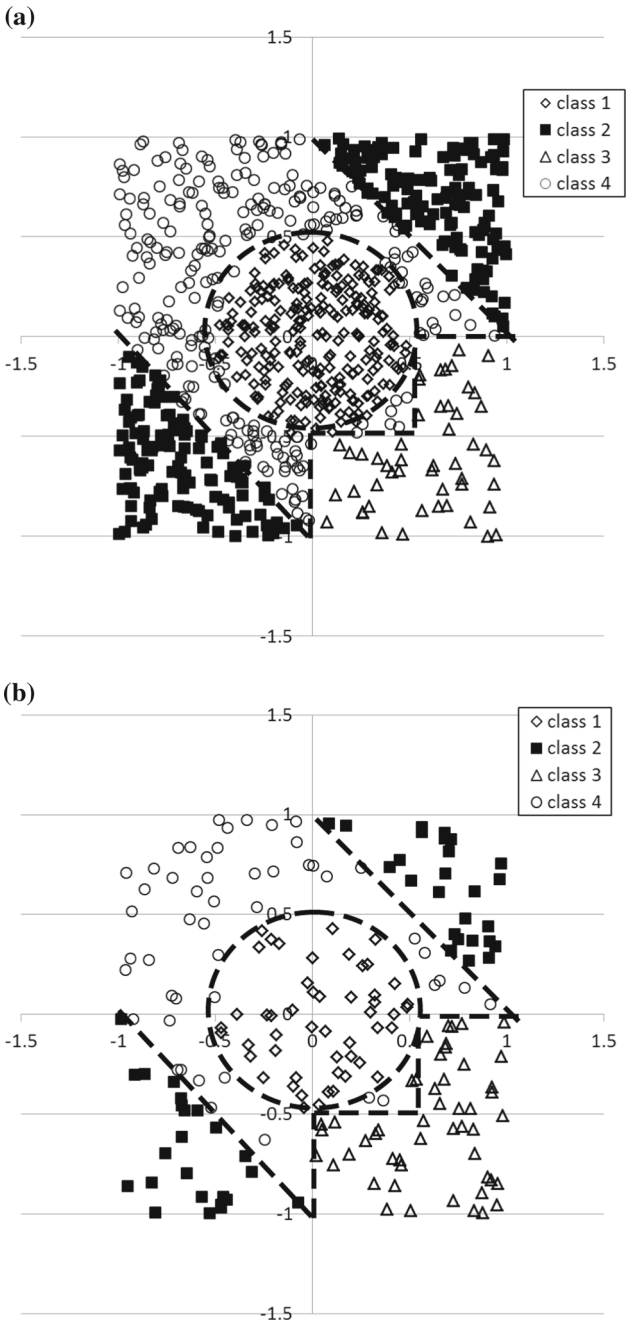

Fig. 7 Artificial *UB* data set: unbalanced data classes. **a** Training set. **b** Test set

with the standard EA, indicating that the discriminant was the evolutionary search. Compared to the standard EA, the two co-evolutionary algorithms obtained at least comparable learning accuracies at much reduced computational costs.

The two co-evolutionary methods gave comparable learning accuracies on all classification benchmarks. However, CANNT was faster than CENNT to reduce the size of the training subsets, and this translated in lower computational overheads. On the artificial *N*00 data set, the competitive

Table 8 Unbalanced data classes—accuracy results

ARTIFICIAL SET *UB* (UNBALANCED DATA SETS)				
	BP	EA	CENNT	CANNT
min	91.50	94.50	94.50	94.00
10th percentile	93.30	94.50	94.95	94.45
median	94.50	95.00	95.75	**96.25**
90th percentile	95.00	95.50	98.10	98.00
max	95.00	95.50	99.00	98.00

(a) overall classification accuracy

ARTIFICIAL SET *UB* (UNBALANCED DATA SETS)				
	BP	EA	CENNT	CANNT
Class1	98.00	**100.00**	98.00	98.00
Class2	99.00	**100.00**	**100.00**	**100.00**
Class3	82.00	82.00	**89.00**	87.00
Class4	99.00	98.00	98.00	**100.00**

(b) breakdown of classification accuracy

Table 9 Unbalanced data classes—confusion matrices, each class in the test set is composed of 50 examples

	Class 1	Class 2	Class 3	Class 4
(a) BP rule—confusion matrix				
Class 1	49.00	0.00	0.00	1.00
Class 2	0.00	49.50	0.00	0.50
Class 3	0.00	0.00	41.00	9.00
Class 4	0.00	0.00	0.00	49.50
(b) Standard EA—confusion matrix				
Class 1	50.00	0.00	0.00	0.00
Class 2	0.00	50.00	0.00	0.00
Class 3	0.00	0.00	41.00	9.00
Class 4	0.00	1.00	0.00	49.00
(c) CENNT—confusion matrix				
Class 1	49.00	0.00	0.00	1.00
Class 2	0.00	50.00	0.00	0.00
Class 3	0.00	0.00	44.50	5.50
Class 4	0.00	1.00	0.00	49.00
(d) CANNT—confusion matrix				
Class 1	49.00	0.00	0.00	1.00
Class 2	0.00	50.00	0.00	0.00
Class 3	0.00	0.00	43.50	6.50
Class 4	0.00	0.00	0.00	50.00

procedures were between three (CENNT) and six (CANNT) times faster than the standard EA. Overall, the CANNT algorithm appears to be preferable to CENNT due to the smaller computational overheads.

A second contribution was made demonstrating the promise of competitive co-evolution for the handling of unbalanced data sets. In Sect. 7, the superior performance of competitive learning over a standard EA and the BP rule was demonstrated on an artificial data set. In particular, the MLPs trained using the co-evolutionary algorithms learned to categorise with superior accuracy the examples of the under-represented class. It is hypothesised that the superior

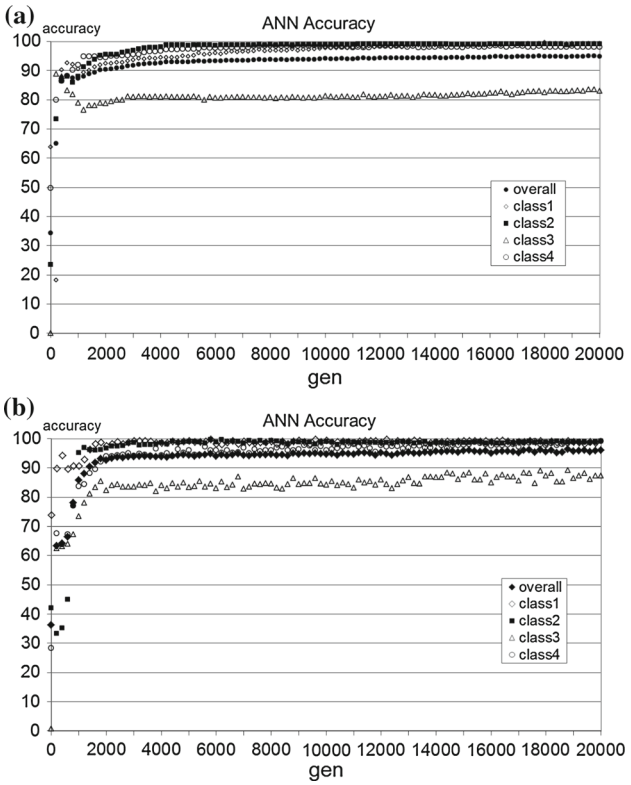

Fig. 8 Unbalanced *UB* data sets, learning curves. **a** EA. **b** CANNT

performance of competitive learning is due to the combination of two strengths: the ability to reduce the contribution of the least important training examples (cost-sensitive learning, boosting), and the ability to alter the data distribution by removing examples (under-sampling).

The procedures used in this study were not customised for the particular kinds of EA and ANN used. The results should be therefore of general validity for the whole field of co-evolutionary ANN classifier learning.

9 Conclusions

This paper discussed the application of co-evolutionary predator–prey approaches for MLP classifier training. Two alternatives evaluated were presented: the first procedure (CANNT) features an evolving population of classifiers (the predators) and a co-adapting population of training examples (the prey) and the second procedure (CENNT) features two true co-evolving populations of classifiers (the predators) and training subsets (the prey).

Experimental evidence clearly indicates the advantages of the competitive evolutionary approach to MLP training in terms of classification accuracy and learning speed. It is hoped that these results will foster the interest towards this yet relatively unexplored approach.

Acknowledgements This study was not funded by any research grant.

Compliance with ethical standards

Conflict of interest The author declares that he has no conflict of interest.

Human and animal rights This article does not contain any studies with human or animal participants performed by the author.

References

Aboitiz F (1992) Mechanisms of adaptive evolution Darwinism and Lamarckism restated. Med Hypotheses 38(3):194–202

Angeline PJ, Pollack JB (1993) Competitive environments evolve better solutions for complex tasks. In: Forrest S (ed) Proceedings of the 5th international conference on genetic algorithms. San Francisco, USA, pp 264–270

Lichman M (2013) UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. University of California, School of Information and Computer Science, Irvine, CA

Azzini A, Tettamanzi AGB (2011) Evolutionary ANNs: a state of the art survey. Intell Artif 5(1):19–35

Castellani M (2013) Evolutionary generation of neural network classifiers: an empirical comparison. Neurocomputing 99:214–229

Chandra R (2013) Memetic cooperative coevolution of Elman recurrent neural networks. Soft Comput 18(8):1549–1559

Chawla NV, Japkowicz N, Kotcz A (2004) Editorial: special issue on learning from imbalanced data sets. ACM SIGKDD Explor Newsl 6(1):1–6

de Boer FK, Hogeweg P (2012) Co-evolution and ecosystem based problem solving. Ecol Inform 9:47–58

Floreano D, Nolfi S (1997) God save the red queen! Competition in co-evolutionary robotics. In: Evolutionary computation, 5

Fogel DB (2000) Evolutionary computation: toward a new philosophy of machine intelligence, 2nd edn. IEEE Press, New York

Fogel DB (2002) Blondie24: Playing at the Edge of AI. Morgan Kaufmann, San Francisco

García-Pedrajas N, Hervás-Martínez C, Muñoz-Pérez J (2003) COVNET: a cooperative coevolutionary model for evolving artificial neural networks. IEEE Trans Neural Netw 14(3):575–596

Gomez F, Schmidhuber J, Miikkulainen R (2008) Accelerated neural evolution through cooperatively coevolved synapses. J Mach Learn Res 9:937–965

Haykin S (2009) Neural networks and learning machines, 3rd edn. Prentice Hall, New York

Hillis WD (1990) Co-evolving parasites improve simulated evolution as an optimization procedure. Phys D 42(1):228–234

Lippmann RP (1987) An introduction to computing with neural nets. IEEE ASSP Mag 4(2):4–22

Mitchell M (2006) Coevolutionary learning with spatially distributed populations. In: Yen GY, Fogel DB (eds) Computational intelligence: principles and practice. IEEE Press, Piscataway, pp 137-154

Mitchell M, Thomure MD, Williams NL (2006) The role of space in the success of coevolutionary learning. In: Rocha LM (ed) Artificial life X: proceedings of the tenth international conference on the simulation and synthesis of living systems. MIT Press, Cambridge, pp 118–124

Nolfi S (2012) Co-evolving predator and prey robots. Adapt Behav 20(1):10–15

Nolfi S, Floreano D (1998) Coevolving predator and prey robots: do "arms races" arise in artificial evolution? Artif Life 4(4):311–335

Paredis J (1998) Coevolutionary process control. In: Artificial neural nets and genetic algorithms. Springer, Vienna, pp 579–582

Paredis J (1995) Coevolutionary computation. Artif Life 2(4):355–375

Paredis J (1996) Coevolutionary life-time learning. In: Rechenberg I, Schwefel HP (eds) Parallel problem solving from nature—PPSN IV. Springer, Berlin, pp 72–80

Pham DT, Castellani M (2010) Adaptive selection routine for evolutionary algorithms. J Syst Control Eng 224(16):623–633

Pollack JB, Blair AD (1998) Co-evolution in the successful learning of backgammon strategy. Mach Learn 32(3):225–240

Popovici E, Bucci A, Wiegand RP, De Jong ED (2012) Coevolutionary principles. In: Rozenberg G et al (eds) Handbook of natural computing. Springer, Berlin, pp 987–1033

Potter MA, De Jong KA (2000) Cooperative coevolution: an architecture for evolving coadapted subcomponents. Evolut Comput 8(1):1–29

Rajagopalan P, Rawal A, Miikkulainen R (2010) Emergence of competitive and cooperative behavior using coevolution. In: Proceedings of the 12th annual conference on genetic and evolutionary computation (GECCO'10). USA, ACM Press, Portland, USA, pp 1073–1074

Ray T (1991) An approach to the synthesis of life. In: Langton C et al (eds) Artificial life II, vol XI. Addison-Wesley Publishing Company Inc, Reading, pp 371–408

Rivera AJ, García-Domingo B, Del Jesus MJ, Aguilera J (2013) Characterization of concentrating photovoltaic modules by cooperative competitive radial basis function networks. Expert Syst Appl 40(5):1599–1608

Rumelhart D, McClelland J (1986) Parallel distributed processing: exploration in the microstructure of cognition, 1–2. MIT Press, Cambridge

Thierens D, Suykens J, Vanderwalle J, De Moor B (1993) Genetic weight optimisation of a feedforward neural network controller. In: Albrecht RF, Reeves CR, Steele NC (eds) Artificial neural networks and genetic algorithms. Springer, Wien, pp 658–663

Uchibe E, Asada M (2006) Incremental coevolution with competitive and cooperative tasks in a multirobot environment. Proc IEEE 94(7):1412–1424

Watson RA, Pollack JB (2001) Coevolutionary dynamics in a minimal substrate. In: Lee S (ed) Proceedings of the genetic and evolutionary computation conference (GECCO 2001), San Francisco, USA. Morgan Kaufmann Publishers, San Francisco, pp 702–709

Weiss GM (2004) Mining with rarity: a unifying framework. ACM SIGKDD Explor Newsl 6(1):7–19

Whitley D (1989) The genitor algorithm and selection pressure: why rank-based allocation of reproductive trails is best. In: Schaffer JD (ed) Proceedings of the third international conference on genetic algorithms, San Mateo, CA. Morgan Kaufmann Publishers, San Francisco, pp 116–123

Yao X (1999) Evolving artificial neural networks. Proc IEEE 87(9):1423–1447