Unifying Learning with Evolution Through Baldwinian Evolution and Lamarckism: A Case Study

Christophe Giraud-Carrier
Department of Computer Science, University of Bristol
The Merchant Venturers Building, Woodland Road, Bristol BS8 1UB, UK
Phone: +44-117-954-5145, Fax: +44-117-954-5208
email: cgc@cs.bris.ac.uk

ABSTRACT: Baldwinian evolution and Lamarckism hold that behaviour is not evolved solely through crossover and mutation, but also through learning. In the former case, learned behaviour causes changes only to the fitness landscape, whilst in the latter, learned behaviour also causes changes to the parents' genotypes. Although the biological plausibility of these positions remains arguable, they provide a potentially useful framework for the construction of artificial systems. As an example, I show the use and effect of Baldwinian and Lamarckian evolution in the design of the hidden layer of a RBF network. Here, evolution is used to optimise the k-means clustering process by co-evolving the two determinant parameters of the network's layout (i.e., the number of centroids and the centroids' positions).

KEYWORDS: Evolutionary Neural Networks, RBF Networks, Baldwin Effect, Lamarckian Learning, Fuzzy Prototypes

INTRODUCTION

The traditional genetic algorithm finds its motivation in Darwin's theory of evolution, whose basic tenets can be summarised as follows.

> During reproduction, the traits found in parents are passed onto their offsprings.
> Individuals survive based on their ability to adapt to the pressures of their environment, so that individuals better suited to the environment tend to have more offsprings and thus drive the population towards favourable traits.
> The traits of offsprings are partially inherited from their parents and partially the result of new traits created through random mutations.

In Darwinian evolution, there is no provision for traits acquired/learned by the individuals of one generation to be passed on to those of the next. In other words, learning does not play any significant role in the evolutionary process. At least two positions have been put forward, which show how learning and evolution can work together synergistically.

The first was advanced by Lamarck (1815), when he argued that "All which has been acquired by, laid down, or changed in the organization of individuals in the course of their life is conserved by generation and transmitted to the new individuals which proceed from those which have undergone those changes." In other words, traits passed from one generation to the next consist of both innate and acquired characteristics of the parents. The other position, somewhat middle ground, was advanced by Baldwin (1896) and has become known as the Baldwin effect. Baldwin's idea was that the fitness value of the improved individual could be transferred to the individual, without actually changing its genotype. In other words, the fitness landscape is allowed to change (i.e., parents are selected based on their fitness after learning), but the evolutionary mechanisms remain Darwinian.

Aside from the plausibility arguments - which are beyond the scope of this paper - there are some practical problems where the Baldwin effect and Lamarckism can prove useful. In fact, most approaches to evolutionary neural networks make implicit use of the Baldwin effect since fitness is generally computed after weight training - an accepted form of learning. It is clear that the alternative, which would compute the fitness of the randomly initialised networks, is unlikely to give very informed results. Work by Gruau and Whitley (1993) has shown that both the Baldwin effect and Lamarckism produce improvement over standard evolution in the cellular development of neural networks. A systematic

study by Whitley, Gordon and Mathias (1994), under the name of hybrid genetic algorithms [1], showed that for binary-encoded function optimisation problems, Lamarckian and Baldwinian evolutions were also superior.

In this paper, we study the use and effect of Baldwinian and Lamarckian learning on the evolution of RBF networks. Burdsall and Giraud-Carrier (1997b) describe an evolutionary strategy, called GA-RBF, that optimises the k-means clustering process by co-evolving the two determinant parameters of the network's layout (i.e., the number of centroids and the centroids' positions). The strategy is already Baldwinian since fitness is computed after k-means clustering. Here, we view the result of k-means clustering, namely the improved positions of the centroids, as the learned "trait". To introduce Lamarckism, we only need to code the new centroids' positions back onto the individuals of the current generation prior to genetic recombination. One of the additional features of GA-RBF is that it does not require the construction of a RBF network to compute its fitness. Instead, it relies on a computationally efficient approximation of the network's execution based on an extension of the nearest-neighbour classification algorithm to fuzzy prototypes with infinite support, developed by Burdsall and Giraud-Carrier (1997a).

The paper is organised as follows. We first give an overview of GA-RBF. We then present the results of experiments on some classical problems. Finally, we conclude with some pointers to related work and a summary of our contributions.

GA-RBF

In this section, we review our previous work on GA-RBF, originally described in Burdsall and Giraud-Carrier (1997b). GA-RBF is an evolutionary system for the automatic design of the hidden layer of a RBF network. The system co-evolves the number of centroids per class and the positions of each centroid in its class. It makes use of the Baldwin effect through a combination of k-means clustering and an extension of the nearest-neighbour classification algorithm to fuzzy prototypes with infinite support.

GENETIC ENCODING

Because centroids are points in a hyper-dimensional real space, a real-valued, rather than a binary-encoded, genetic algorithm (GA) is used, causing the representation to be of a more phenotypical nature. Each chromosome, or individual, encodes one set of centroids starting positions for each class, as follows.

$$individual = [\{p_1^1,..., p_{k_1}^1\},\{p_1^2,..., p_{k_2}^2\},...,\{p_1^n,..., p_{k_n}^n\}]$$

where n is the number of classes and k_i is the number of centroids for class i. The k_i's vary continually during evolution, thus providing the necessary diversity in the population to optimise both determinant parameters. In the current implementation, a practical upper bound of 7 is placed on the k_i's since empirical evidence suggests that most problems require fewer than 7 centroids per class.

FITNESS CALCULATION

RBF networks learn from a training set and subsequently generalise to previously unseen instances. In order to avoid overfitting, GA-RBF's training set is split into a clustering set and an evaluation set. The objective fitness function used for evaluating individuals consists of the application of the k-means algorithm to the clustering set using the starting positions encoded in the individual's genes, followed by a test classification of the evaluation set using the k-means-computed centroids and the nearest-attracting prototype (NAP) classifier introduced by Burdsall & Giraud-Carrier (1997a). Hence,

$$fitness = \frac{number\,of\,correct\,classifications}{size\,of\,evaluation\,set} \times 100$$

The NAP classifier extends the classical nearest-neighbour classifier by replacing the traditional Euclidean distance with an infinite fuzzy support membership function as in RBF networks. Using NAP classifiers rather than RBF networks saves the construction of the networks and the associated computation of the output weights at each generation of the

[1] According to Whitley, Gordon and Mathias (1994), a "hybrid genetic algorithm" combines local search with a more traditional genetic algorithm, that is, it extends the genetic algorithm with either Baldwinian or Lamarckian learning. Most evolutionary neural network strategies, since they use the Baldwin effect, can thus be regarded as instances of hybrid genetic algorithms.

GA. Hence, the NAP classifier is used here as a computationally efficient approximation of a RBF network's output layer's performance. Note that although the centroids are labelled during optimisation, the labels would be removed once they were placed in the RBF network upon convergence. The early labelling allows GA-RBF to use output information and to conduct a faster, more informed search for an optimal solution.

The Baldwin Effect and Lamarckism

As mentioned above, calculating fitness involves two steps, namely k-means clustering and NAP classification. The effect of k-means clustering is to refine the starting positions of the centroids to more "representative" final positions. At the individual's level, this may be viewed as a form of learning, since NAP classification based on the final centroids' positions is most likely to yield better results than NAP classification based on their starting positions. Hence, through k-means clustering, an individual improves its performance.[2] As fitness is computed after learning, GA-RBF makes implicit use of the Baldwin effect. If, in addition, the final centroids' positions are coded back into the individuals' chromosomes, then Lamarckian evolution can also be achieved.

REPRODUCTION

The selection of individuals for reproduction is biased towards fitter individuals, as follows. Let N be the size of the population and *fitness(i)* denote the fitness of individual i. The probability that i is selected is:

$$p_i = \frac{fitness(i)}{\sum_{j=1}^{N} fitness(j)}$$

The p_i's define a probability density function over the population. The parents chosen for reproduction are obtained by sampling the distribution without replacement. Genetic operators are applied to pairs of parents to produce offsprings that replace the least fit individuals in the population.

Because a phenotypic representation is used, the traditional genetic operators have to be modified to maintain the integrity of the solution space. The classical crossover operator is replaced by a novel operator, called gene pooling, and three different types of mutation are introduced.

Gene Pooling

Gene pooling takes two parents and produces a single offspring. Following Fogel (1994), a population is viewed more like a pool of genes than a collection of individuals. A gene, here, is a single centroid's position. During crossover the genes of the two parents are mixed together in a kind of genetic soup. From this mixture, a new individual is formed by random selection of genes, as follows.

$$\begin{aligned} parentA &= \vec{A}_1^1,..., \vec{A}_{a_1}^1, \vec{A}_1^2,..., \vec{A}_{a_2}^2,..., \vec{A}_1^n,..., \vec{A}_{a_n}^n \\ parentB &= \vec{B}_1^1,..., \vec{B}_{b_1}^1, \vec{B}_1^2,..., \vec{B}_{b_2}^2,..., \vec{B}_1^n,..., \vec{B}_{b_n}^n \end{aligned} \Rightarrow childAB = \vec{C}_1^1,..., \vec{C}_{c_1}^1, \vec{C}_1^2,..., \vec{C}_{c_2}^2,..., \vec{C}_1^n,..., \vec{C}_{c_n}^n$$

where: - C_i^j is a random number generated from a normal distribution centred on $C_i^j \neq C_k^j$

-is drawn at random from $\{\vec{A}_1^j,..., \vec{A}_{a_i}^j, \vec{B}_1^j,..., \vec{B}_{b_i}^j\}$

-for all $i \neq k$ $\frac{a_i + b_i}{2}$

The first condition causes the child to have a length roughly the average of the lengths of its parents. The second condition ensures that genes of a class in the child are drawn from genes of the same class in the parents (i.e., gene pooling does not disrupt the class memberships of the centroids). Finally, the third condition ensures that there are no

[2] Simon (1983) defines learning as "any change in a system that allows it to *perform better* the second time on repetition of the same task or another task drawn from the same population." (emphasis added).

duplicates. Gene pooling is a valid form of crossover since, in the adopted representation, the order of genes on a chromosome is irrelevant.

Mutation

Mutation is applied to offsprings and consists of adding, removing or swapping a centroid in a randomly chosen class in the chromosome. Let K be the maximum number of genes per class (here, K=7). Let i be the randomly selected class and k_i be the number of genes in class i. The probabilities of occurrence of each mutation are given by:

$$P_{add} = \frac{k_i}{K} \times g_{add}$$
$$P_{rem} = \frac{K - k_i}{K} \times g_{rem}$$
$$P_{swap} = 0.3$$

where γ_{add}=0.67 and γ_{rem}=0.875. Mutation by adding and mutation by removing are generally mutually exclusive. Together with gene pooling, they provide a way of optimising the number of centroids per class. The centroids added or swapped during mutation are drawn at random from within the clustering data points for the selected class. They may not already exist in the class.

POPULATION CONTROL

In addition to the above, GA-RBF uses a punishment function to combat over-fitting and an ageing function to prevent saturation by super-individuals.

Punishment

Individuals with more centroids than others generally have higher fitness as they can place their numerous centroids near the training examples. Unfortunately, these individuals also tend to over-specialise on the evaluation set used and to perform poorly on other sets, i.e., over-fitting occurs. To prevent over-fitting, GA-RBF allows individuals to evolve larger number of centroids per class but punishes them for doing so. Here, the punishment introduces a form of natural handicap for highly fit, greedy individuals. It is given by:

$$punishment = \sum_{i=1}^{n} \alpha_i \times \frac{1.38}{n} \times (k_i - 3) \times fitness(i)$$

where n, k_i and $fitness(i)$ are as before, and α_i is 1 if k_i>3 and 0 otherwise. Only greedy individuals are punished and their punishment increases with both greed and fitness. With the addition of the punishment mechanism, the actual fitness function for GA-RBF becomes:

$$fitness' = fitness - punishment$$

Ageing

Individuals whose fitness far exceed the average fitness of the population are called super-individuals. Super-individuals reproduce rapidly and come to dominate the population after only a few generations. The propagation of super-individuals leads to rapid exploitation but often results in premature convergence to a local optimum. A reasonable amount of exploration must be maintained. To address this issue, the notion of a life span for individuals has been suggested by Michalewicz (1992). GA-RBF follows this idea and defines the life span of individual i, at "birth", by:

$$lifespan(i) = b \times \frac{fitness(i)}{average\ population\ fitness}$$

where β is some constant. Each time an individual reproduces, its life span is reduced by 1. When the life span reaches 0 the individual dies and is removed from the population. Should a parent die following reproduction, its offspring takes its place rather than that of the weakest individual in the population.

EXPERIMENTAL RESULTS

Two datasets from Blake and Merz (1998) were originally used to test GA-RBF. The noisy versions contain the original data with substantial noise added. Multiple simulations and cross-validation techniques were used to increase the validity of the results. The results for GA-RBF are compared against those of a standard NAP classifier with 3 (k-means-generated) centroids per class, an user-optimised infinite RBF network with 3 centroids per class, and an optimised multilayer perceptron (MLP). Results are in Table I and report predictive accuracy on unseen test data.

Application	***GA-RBF***	***RBF***	***NAP***	***MLP***
Iris	96.3	96.3	95.6	96.0
Iris (noise)	84.4	81.7	79.9	83.0
Diabetes	95.6	95.1	95.2	96.0
Diabetes (noise)	89.6	89.0	86.3	85.1
Averages	**91.5**	**90.5**	**89.3**	**90.0**

Table I: GA-RBF Performance

The fixed number 3 of centroids per class used in the non-evolutionary models results from intensive, user-driven testing. With GA-RBF, the number of centroids per class evolves naturally to settle on a value between 2 and 4. The cost of the evolution process is relatively low as convergence occurs in less than 3 minutes (Sun Workstation). Moreover, with no a priori knowledge nor manual fine-tuning, GA-RBF results in a slight increase of predictive accuracy for the problems considered.

The results so far only make (implicit) use of the Baldwin effect. Further experiments with additional applications and with Lamarckian evolution are under way.

DISCUSSION AND CONCLUSION

The system discussed here applies evolution to the design of the hidden layer of a RBF network, through optimisation of the k-means clustering procedure and an efficient approximation of the output layer's execution based on nearest-attracting fuzzy prototypes. In addition to being implicitly Baldwinian, evolution can be Lamarckian by coding the centroids found by k-means back onto the chromosomes of the population prior to genetic recombination. Hence, the system combines learning with evolution and neural networks.

The past decade has seen an explosion of the work on evolutionary neural networks, targeted mostly at automating the setting of parameters in feedforward neural networks (e.g., see the work of Mühlenbein and Kinderman (1989), the survey of Schaffer et al (1992), the review of Yao (1993), the general framework proposed by Yao and Liu (1998), and the increasing number of papers and workshops addressing these issues). The work closest to ours, since it also considers RBF networks, is that of Neruda (1995), in which functionally-equivalent canonical parametrisations of RBF networks are evolved.[3]

Considering the use of Lamarckian evolution with neural networks, previous experiments with networks using cellular encoding are reported in Gruau and Whitley (1993) and with networks using marker-based encoding in Pederson (1996). To our knowledge, this work is the first to report results on the use of Lamarckian evolution in the context of RBF networks and using a real-valued encoding. Furthermore, we have established a conceptual connection between evolutionary neural networks and hybrid genetic algorithms by showing that, in most instances, the former, as they make implicit use of the Baldwin effect, are only an instance of the latter.

[3]To keep the discussion within the context of evolution and neural networks, we purposely ignore references to other approaches to k-means optimisation. Several such references may be found in Burdsall and Giraud-Carrier (1997b).

REFERENCES

Baldwin, J.M., 1896, "A New Factor in Evolution", American Naturalist, **30**:441-451.

Blake, C.L.; Merz, C.J., 1998, "UCI Repository of Machine Learning Databases", University of California, Irvine/CA, Department of Information and Computer Sciences. (http://www.ics.uci.edu/~mlearn/MLRepository.html).

Burdsall, B.; Giraud-Carrier, C., 1997a, "Evolving Fuzzy Prototypes for Efficient Data Clustering", Second International ICSC Symposium on Fuzzy Logic and Applications, Zürich, Switzerland, 217-223.

Burdsall, B.; Giraud-Carrier, C., 1997b, "GA-RBF: A Self-optimising RBF Network", Third International Conference on Artificial Neural Networks and Genetic Algorithms, Norwich, UK, 346-349.

Fogel, L.J., 1994, "Evolutionary Programming in Perspective: The Top Down View", Computational Intelligence: Imitating Life, Zurada, J.M.; et al., (Eds.), IEEE Press, Inc., NY.

Gruau, F.; Whitley, D., 1993, "Adding Learning to the Cellular Developmental Process: A Comparative Study", Evolutionary Computing, **1**(3):213-233.

Lamarck, J.B., 1815, "Histoire Naturelle des Animaux Sans Vertèbres", (in French). Translated and published as "Zoological Philosophy: An Exposition with Regard to the Natural History of Animals", 1984, University of Chicago Press, Chicago/IL, USA.

Michalewicz, Z., 1992, "Genetic Algorithms + Data Structures = Evolution Programs", Springer-Verlag.

Mühlenbein, H.; Kinderman, J., 1989, "The Dynamics of Evolution and Learning - Toward Genetic Neural Networks", Connectionism in Perspective, Pfeifer, R.; et al.., (Eds.), Elsevier.

Neruda, R., 1995, "Functional Equivalence and Genetic Learning", Proceedings of the First International Conference on Artificial Neural Networks and Genetic Algorithms, 53-56.

Pederson, K., 1996, "Evolving Neural Networks with Genetic Algorithms", Project Report PR-96-33, University of Bristol, Department of Computer Science.

Schaffer, J.D.; Whitley, D; Eshelman, L.J., 1992, "Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art", Proceedings of the International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1-37.

Simon, H., (1983), "Why Should Machines Learn?", Machine Learning: An Artificial Intelligence Approach, Vol. 1, Michalski, R.S.; Carbonell, J.G.; Mitchell, T.M., (Eds.), Tioga, Palo Alto/CA.

Whitley, D.; Gordon, V.S.; Mathias, K., 1994, "Lamarckian Evolution, The Baldwin Effect and Function Optimization", Parallel Problem Solving from Nature - PPSN III, Davidor, Y.; Schwefel, H.P.; Manner, R., (Eds.), Springer-Verlag, 6-15.

Yao, X., 1993, "A Review of Evolutionary Artificial Neural Networks", International Journal of Intelligent Systems, **8**(4):539-567.

Yao, X.; Liu, Y., 1998, "Towards Designing Artificial Neural Networks by Evolution", Applied Mathematics and Computation, **91**(1):83-90.