ELSEVIER

BioSystems 65 (2002) 37–47

Bio Systems

www.elsevier.com/locate/biosystems

Quantitative structure–activity relationships by evolved neural networks for the inhibition of dihydrofolate reductase by pyrimidines

Dana G. Landavazo, Gary B. Fogel *, David B. Fogel

Natural Selection Inc., 3333 N. Torrey Pines Ct., Suite 200, La Jolla, CA 92037, USA

Received 18 October 2001; received in revised form 1 November 2001; accepted 2 November 2001

Abstract

Evolutionary computation provides a useful method for training neural networks in the face of multiple local optima. This paper begins with a description of methods for quantitative structure activity relationships (QSAR). An overview of artificial neural networks for pattern recognition problems such as QSAR is presented and extended with the description of how evolutionary computation can be used to evolve neural networks. Experiments are conducted to examine QSAR for the inhibition of dihydrofolate reductase by pyrimidines using evolved neural networks. Results indicate the utility of evolutionary algorithms and neural networks for the predictive task at hand. Furthermore, results that are comparable or perhaps better than those published previously were obtained using only a small fraction of the previously required degrees of freedom. © 2002 Elsevier Science Ireland Ltd. All rights reserved.

Keywords: Artificial neural networks; Quantitative structure–activity relationships; Dihydrofolate reductase; Evolutionary computation

1. Introduction

The effectiveness of drug discovery can be enhanced through the accurate prediction of biological activity or chemical properties of a molecule from a set of structure-based descriptors. A general assumption of this approach is that the activity or property of a molecule is implicit in its physical structure. Quantitative structure–activity relationships (QSAR) are used to develop statistical models relating structure to activity. Following the identification of a relatively small set of related or unrelated molecules with known activities or properties, many descriptors for these molecules are determined. These are grouped generally into three major classes: hydrophobic, electronic, and steric effects, however, several subdivisions of each may be used. Once a suitable statistical model is developed, it can be used to predict the activity of a novel compound and may also provide insight toward the mechanism of action for a particular drug.

Over the past decade, artificial neural networks have been applied on QSAR in the analysis of

* Corresponding author. Tel.: +1-858-455-6449; fax: +1-858-455-1560.

E-mail address: gfogel@natural-selection.com (G.B. Fogel).

0303-2647/02/$ - see front matter
PII: S0303-2647(01)00192-7

2,4-diamino-5-(substituted benzyl)pyrimidines (So and Richards, 1992), 2,4-diamino-6-dimethyl-5-phenyldihydrotriazines as dihydrofolate reductase (DHFR) inhibitors (Andrea and Kalayeh, 1991), and neural networks have been recognized as an important tool in drug discovery (Kovesdi et al., 1999; Schneider, 2000). DHFR catalyzes the NADPH-dependent reduction of 7,8-dihydrofolate (H2F) to 5,6,7,8-tetrahydrofolate (H4F) and is necessary for maintaining intracellular levels of H4F, an essential cofactor in the synthetic pathway of purines, thymidylate and several amino acids. Evolved neural networks for QSAR have also been proposed (So and Karplus, 1996a,b, 1997a,b; So et al., 2000). Here we review the concepts of evolved artificial neural networks for pattern recognition and indicate the effectiveness of evolved artificial neural networks for QSAR by using a small subset of available features to predict activity for a set of 74 2,4-diamino-5-(substituted benzyl)pyrimidines. Relative to previous literature (Hirst et al., 1994), the evolved neural network was capable of similar predictive accuracy to the model given in Hirst et al. (1994) using only three of the 27 indicated input features.

1.1. Artificial neural networks for pattern recognition

Artificial neural networks (or simply *neural networks*) are computer algorithms based loosely on modeling the neuronal interconnections of natural organisms. They are stimulus–response transfer functions that accept some input and yield some output. They are used to typically learn an input–output mapping over a set of examples.

Neural networks are parallel processing structures consisting of nonlinear processing elements interconnected by fixed or variable weights. They are quite versatile, for they can be constructed to generate arbitrarily complex decision regions for stimulus–response pairs. That is, in general, if given sufficient complexity, there exists a neural network that will map every input pattern to its appropriate output pattern, so long as the input–output mapping is not one-to-many (i.e. the same input having varying output). Neural networks are therefore well suited for use as detectors and classifiers. The classic pattern recognition algorithms require assumptions concerning the underlying statistics of the environment. Neural networks, in contrast, are nonparametric and can effectively address a broader class of problems (Lippmann, 1987).

Multilayer perceptrons, also sometimes described as *feed forward networks*, are probably the most common architecture used in supervised learning applications (where exemplar patterns are available for training). Each computational node sums N weighted inputs, where w_{ij} is the weight between nodes i and j, and a_i is the activation of the ith node, subtracts a threshold value, θ_j, and passes the result through a logistic (sigmoid) function, generating the output:

$$f(\beta_j) = (1 - e^{-\beta_j})^{-1},$$

where $\beta_j = (\Sigma\, a_i w_{ij}) - \theta_j,\ i = 1, \ldots, N$. Single perceptrons (i.e. feed forward networks consisting of a single input node) form decision regions separated by a hyperplane. If the input from the given different data classes are linearly separable, a hyperplane can be positioned between the classes by adjusting the weights and bias terms. If the inputs are not linearly separable, containing overlapping distributions, a least mean square (LMS) solution is typically generated to minimize the mean squared error between the calculated output of the network and the actual desired output. While single perceptrons can generate hyperplane boundaries, perceptrons with a hidden layer of processing nodes have been proven to be capable of approximating any measurable function (Hornik et al., 1989), indicating their broad utility for addressing general pattern recognition problems.

Another versatile neural network architecture is the *radial basis function network*. Rather than partitioning the available data using hyperplanes, the radial basis function network clusters available data, often with the use of approximate Gaussian density functions. The network comprises an input layer of nodes corresponding to the input feature dimension, a single hidden layer of nodes with computational properties described below, and output nodes, which perform linear combinations on the hidden nodes. Each connec-

tion between an input and hidden node carries two variable parameters corresponding to a mean and standard deviation. Poggio and Girosi (1990) proved that linear combinations of these near-Gaussian density functions can be constructed to approximate any measurable function. Therefore, like the multilayer perceptron, radial basis functions are universal function approximators.

Given a network architecture (i.e. type of network, the number of nodes in each layer, the connections between the nodes, and so forth), and a training set of input patterns, the collection of variable weights determines the output of the network to each presented pattern. The error between the actual output of the network and the desired target output defines a response surface over an n-dimensional hyperspace, where there are n parameters (e.g. weights) to be adapted. Multilayer feed forward perceptrons are the most commonly selected architecture and training these networks can be accomplished through a *back propagation* algorithm that implements a gradient search over the error response surface for the set of weights that minimizes the sum of the squared error between the actual and target values.

Although, the use of back propagation is common in neural network applications, it is quite limiting. This procedure is mathematically tractable and provides guaranteed convergence, but only to a locally optimal solution. Even if the network's topology provides sufficient complexity to solve the given pattern recognition task completely, the back propagation method may be incapable of discovering an appropriate set of weights to accomplish the task. When this occurs, the operator has several options: (1) accept suboptimal performance; (2) restart the procedure and try again; (3) use ad hoc tricks, such as adding noise to the exemplars; (4) collect new data and retrain; or (5) add degrees of freedom to the network by increasing the number of nodes and/or connections.

Only this last approach, adding more degrees of freedom to the network, is guaranteed to give adequate performance on the training set, provided that sufficient nodes and layers are available. Yet this also presents problems to the designer of the network, for any function can map any measurable domain to its corresponding range if given sufficient degrees of freedom. Unfortunately, such overfit functions generally provide very poor performance during validation on data acquired independently. Such anomalies are encountered commonly in regression analysis, statistical model building, and system identification. Assessing the proper trade-off between the goodness-of-fit to the data and the necessary degrees of freedom requires information criteria (e.g. Akaike's information criterion, minimum description length principle, predicted squared error, or others) (see Fogel, 1991a for a discussion of these criteria). By relying on the back propagation method, the designer almost inevitably accepts that the resulting network will not satisfy the maxim of parsimony, simply because of the defective nature of the training procedure itself. The problems of local convergence with the back propagation algorithm indicate the desirability of training with stochastic optimization methods, such as simulated evolution, which can provide convergence to globally optimal solutions.

1.2. Evolutionary computation and neural networks

Natural evolution is a population-based optimization process. Simulating this process on a computer results in stochastic optimization algorithms that can often outperform classical methods of optimization when applied to difficult real-world problems. Historically, there have been three main avenues of research in evolutionary computation: evolutionary programming (EP), evolution strategies (ES), and genetic algorithms (GA) (Bäck et al., 1997). The methods are broadly similar in that each maintains a population of trial solutions, imposes random changes to those solutions and incorporates the use of selection to determine which solutions to maintain into future generations and which to remove from the pool of trials. The methods used to differ in the types of random changes that were used and the methods for selecting successful trials. Fogel (2000) provided a review of the similarities and differences between these procedures, but indicated the methods are now for all intents and

purposes identical, and there appears to be no scientific rationale for discriminating between 'genetic' and 'evolutionary' computation.

This is particularly true in light of the 'no free lunch' theorem of Wolpert and Macready (1997), which in broad terms states that all algorithms that do not resample points in a search space perform the same on average when applied across all possible functions. Thus, no choice of search operator (e.g. crossover or mutation) can be uniformly superior. The choice must be matched to the problem at hand. Fogel and Ghozeil (1997) offered a related result indicating that there is no advantage to any bijective mapping (representation) on any problem: a change in representation can be offset by a change in variation operator to generate an equivalent algorithm.

Each of the characteristics that made the previously different approaches to simulating evolution unique has been shown to be without any basis for any general mathematical advantage. Moreover, much of the early mathematical foundations of 'schema processing' have recently been shown to be in error (Macready and Wolpert, 1998; see also Fogel and Ghozeil, 1997) or not relevant for optimization (Altenberg, 1995; Vose and Wright, 2001). Rather than seek differences between 'genetic' and 'evolutionary' computation, a more effective approach appears to be to tailor the representation, variation operators, and selection procedures to the optimization task at hand. Evolutionary computation offers a broad range of choices for each of these considerations.

The procedures generally proceed as follows. A problem to be solved is cast in the form of an objective function that describes the worth of alternative solutions. Without loss of generality, suppose that the task is to find the solution that minimizes the objective function. A collection (population) of trial solutions is selected at random from some feasible range across the available parameters. Each solution is scored with respect to the objective function. The solutions (parents) are then mutated and/or recombined with other solutions in order to create new trials (offspring). These offspring are also scored with respect to the objective function and a subset of the parents and offspring are selected to become parents of the next iteration (generation) based on their relative performance. Those with superior performance are given a greater chance of being selected than are those of inferior quality. Fogel (2000) details examples of evolutionary algorithms applied to a wide range of problems, including breast cancer detection (both from histologic and radiographic data), pharmaceutical design, and a variety of industrial settings.

Evolutionary computation has been demonstrated as a useful means for training neural networks, particularly as compared to gradient-based methods such as back propagation. For example, Porto et al. (1995) compared back propagation, simulated annealing and evolutionary programming for training a fixed network topology to classify active sonar returns. The results indicated that stochastic search techniques such as annealing and evolution outperform back propagation consistently, yet can be executed more rapidly on an appropriately configured parallel processing computer. After sufficient computational effort, the most successful network can be put into practice. But the evolutionary process can be continued during application, so as to provide iterative improvements on the basis of newly acquired exemplars. The procedure is efficient because it can use the entire current population of networks as initial solutions to accommodate each newly acquired datum. There is no need to restart the search procedure in the face of new data, in contrast with many classic search algorithms, such as dynamic programming.

Designing neural networks through simulated evolution follows an iterative procedure:

1. A specific class of neural networks is selected. The number of input nodes corresponds to the amount of input data to be analyzed. The number of classes of concern (i.e. the number of classification types of interest) determines the number of output nodes.
2. Exemplar data is selected for training.
3. A population of P complete networks is selected at random. A network incorporates the number of hidden layers, the number of nodes in each of these layers, the weighted connections between all nodes in a feedforward or other design, and all of the bias terms associ-

ated with each node. Reasonable initial bounds must be selected for the size of the networks, based on the available computer architecture and memory.

4. Each of these 'parent' networks is evaluated on the exemplar data. A payoff function is used to assess the worth of each network. A typical objective function is the root mean-squared error (RMSE) between the target and the actual output summed over all output nodes; this technique is often chosen because it simplifies calculations in the back propagation training algorithm. As evolutionary computation does not rely on similar calculations, any arbitrary payoff function can be incorporated into the process and can be made to reflect the operational worth of various correct and incorrect classifications. Furthermore, information criteria such as Akaike's information criterion (AIC) (Fogel, 1991b) or the minimum description length principle (Fogel and Simpson, 1993) can provide mathematical justification for assessing the worth of each solution, based on its classification error and the required degrees of freedom.
5. 'Offspring' are created from these parent networks through random mutation. Simultaneous variation is applied to the number of layers and nodes, and to the values for the associated parameters (e.g. weights and biases of a multilayer perceptron, weights, biases, means and standard deviations of a radial basis function network). A probability distribution function is used to determine the likelihood of selecting combinations of these variations. The probability distribution can be preselected a priori by the operator or can be made to evolve along with the network, providing for nearly completely autonomous evolution (Fogel, 2000).
6. The offspring networks are scored in a similar manner as their parents.
7. A probabilistic round-robin competition is conducted to determine the relative worth of each proposed network. Pairs of networks are selected at random. The network with superior performance is assigned a 'win'. Competitions are run to a preselected limit. Those networks with the most wins are selected to become parents for the next generation. In this manner, solutions that are far superior to their competitors have a corresponding high probability of being selected. The converse is also true. This function helps prevent stagnation at local optima by providing a parallel biased random walk.
8. The process iterates by returning to step (5).

1.3. Application of evolved neural networks for quantitative structure–activity relationships

Neural networks are useful for QSAR because there are many potential parameters for each molecule and the contribution of each parameter is not known apriori. The number of input parameters (and their combination) is sufficiently large such that an exhaustive search is not computationally feasible. One method to reduce this dimensionality is to make assumptions about the most-relevant parameters on the basis of some information criterion or other measure. However, this type of reduction may lead to convergence on local minima. Evolutionary computation can be used as a tool to rapidly search for the appropriate number of input parameters while avoiding premature convergence due to inappropriate heuristics.

An early attempt for evolved neural networks in QSAR was offered in Tetko et al. (1994), which used EP for structure–activity relationships (SAR). QSAR methods focus on quantitative values to describe biological activity, whereas SAR methods focus on broader classifications of activities for prediction of overall worth. In their work, EP was used to search for parameter sets with the best expected error of activity classification. Tetko et al. (1994) demonstrated that EP could be used to anticipate the hypoglycemic activity of flavonoids using a set of 52 molecules divided into two groups: active and non-active. Ninety-three parameters were used for each molecule (electrostatic, topologic and spatial characteristics).

At nearly the same time, Luke (1994) offered a method for the use of EP for QSAR. As many as 53 parameters were used for 31 analogues of antifilarial antimycin as reported from a data set

in the literature. A population of N predictors was generated at random. The fitness for each predictor (with respect to a payoff function described in the article) was calculated. Each predictor was allowed to create a new predictor. The $2N$ predictors were ranked and the n least-fit predictors were removed from the population, restoring the population size to N. This process was iterated for a set number of generations. In his analysis, Luke noted that 'in this study, evolutionary programming [was] chosen over a genetic algorithm for the simple reason that the latter may have problems when the number of descriptors increases in a QSAR/QSPR (quantitative structure–property relationship) investigation. Though the … data has 53 descriptors, it is not hard to imagine a set of data containing 1000 descriptors. If a researcher wants to find good predictors that only use three of the descriptors ($n$term = 3) the n_i vector contains 997 zeroes and three nonzero values. If both parents have $n$term = 3, it is very likely that $n$term will not be 3 in an offspring … in a standard genetic algorithm, the offspring … replaces the least fit predictor in the set … whether or not the offspring's fitness is better than the solution it is producing. Therefore, there is no guarantee that the overall fitness of the population will increase from generation to generation in the case where a large number of descriptors are present' (p. 1281). Four data sets were used in this study, each of which was borrowed from other sources in the literature.

Following this work, So and Karplus (1996a,b) investigated both EP and GAs for the training of neural nets for QSAR on the same initial data set as Luke (1994) using 53 parameters for 31 antimycin candidates. Their results suggested that, although both procedures arrived at the best solution in under 10 generations, the average fitness of the population was significantly higher when using an EP approach. So and Karplus used only the EP algorithm in the remainder of their analysis. This same approach was applied to a homogeneous set of 57 classical 1,4-benzodiazepin-2-ones with binding affinities for the BZ/$GABA_A$ receptor. The data set used for this investigation was reported elsewhere in the literature and in this study. Once again, the authors used EP approach to show 'the general utility of the genetic neural network (GNN) methodology in dealing with data sets of high dimensionality' (p. 5255). So and Karplus extended their work with two additional papers in 1997 regarding 3D QSAR with 'genetic' neural networks (even though the procedure employed was in fact more akin to EP).

2. Methods and results

For the experiments presented here, data on the biological activities for 55 2,4-diamino-5-(substituted benzyl)pyrimidines were used (Hirst et al., 1994). Biological affinities were measured by the association constant ($\log K_i$) to DHFR from *Escherichia coli* MB1428. These data were used to test the performance of neural networks trained using backpropagation versus those trained using evolutionary computation. Hirst et al. (1994) divided these data into five sets of 11 molecules each. Using leave-one-out cross-validation (Efron, 1982), each of these data sets was used as a test set with the other four sets serving as training sets. Therefore, each of the 55 pyrimidines appeared only once over all of the test sets.

Each of the 55 molecules in the data set contained three positions suitable for replacement (the 3-, 4-, and 5-positions of the phenyl ring) by a different substituent (i.e. –OH, $-O(CH_2)_6CH_3$, etc.) and the activity for each of these substituent combinations was measured. The data reported in Hirst et al. (1994) suggested that addition of hydroxyl groups at positions 3 and 5, and a hydrogen at position 4 produced an activity (3.04), which was significantly below the mean of the remaining data ($\mu = 7.115$, $\sigma = 0.935$). This data point was removed from all training and testing data sets as a significant outlier, despite it being used by Hirst et al. (1994).

All substituents were assigned attributes in Hirst et al. (1994) with regards to polarity, size, flexibility, number of hydrogen-bond donors and acceptors, presence and strength of π-donors and -acceptors, polarizability of the molecular orbitals, the σ-effect, and branching. Therefore, each substituent position was characterized by a vector of 10 numbers containing the attribute

information. It remains unclear from Hirst et al. (1994) whether all 10 of these attributes were used for each of the three positions or if the value for the branching attribute was removed as an input vector to the neural network. Given this uncertainty, we chose to use all 30 attributes describing each of 55 molecules in the database for development of an evolved neural network.

A neural network with a fixed architecture consisting of 30 inputs, three hidden nodes, and one output (as similar as possible to the description provided by Hirst et al., 1994) was used for prediction of activity. For all training sets, a population of 200 parents and 100 offspring was used to evolve the appropriate weights for each connection in the neural network to increase the predictive accuracy over time. Every generation, each parent network architecture generated offspring network architectures by varying all of its weighted connections simultaneously, following a Gaussian distribution with zero mean and an adaptable standard deviation. The update rule for the standard deviation followed the standard method shown in Bäck et al. (1997), Fogel (2000), where lognormal perturbations are applied to the standard deviations before generating offspring. Specifically, offspring were created using the following equations:

$$(1)\quad \boldsymbol{\sigma}'_i = \boldsymbol{\sigma}_i \times \exp(\tau N(0,1) + \tau' N_i(0,1)),$$
$$i = 1, \ldots, n$$

$$(2)\quad \boldsymbol{x}'_i = \boldsymbol{x}_i + \boldsymbol{\sigma}'_i N(0,1), \quad i = 1, \ldots, n$$

where i denotes the ith dimension of the solution vector $\boldsymbol{x}$ or strategy parameter vector $\boldsymbol{\sigma}$, $N(0, 1)$ is a standard Gaussian random variable, $N_i(0, 1)$ designates that a standard Gaussian random variable is sampled anew for each ith dimension, and τ and τ' are constants set equal to $1/\mathrm{sqrt}(2n)$ and $1/(2\mathrm{sqrt}(n))$, respectively, where n is the number of dimensions in $\boldsymbol{x}$ and $\boldsymbol{\sigma}$. The weights of each connection on each member of the initial population were distributed uniformly at random in the range $(-0.5, 0.5)$ and the initial standard deviation in each dimension for each parent was set to 0.05.

Selection was based on each neural network's total RMSE on the training data, with a stochastic tournament being used to determine which parent and offspring networks survived into the next generation. For this tournament, each network was paired up against 10 other randomly selected networks from the population; comparisons in performance were made in each pairing with a 'win' being assigned to the network having lower RMSE. After all such pairings were completed, those networks with the most wins were selected to be parents for the successive generation.

For each evolution, the number of generations was varied from 10 to 200 to determine the most appropriate point to stop neural network training to avoid under- or overfitting of the data. Performances were measured on data held out from training based on the Spearman rank correlation coefficient (following Hirst et al., 1994) and the RMSE. Values for these two performance criteria are shown in Fig. 1. Using the Spearman rank correlation coefficient, the variance in the data is much larger than expected and cannot be used easily for addressing performance. Using the RMSE performance criterion, the measure is minimized at 50 generations and its variance is much smaller over subsequent generations. Given this,

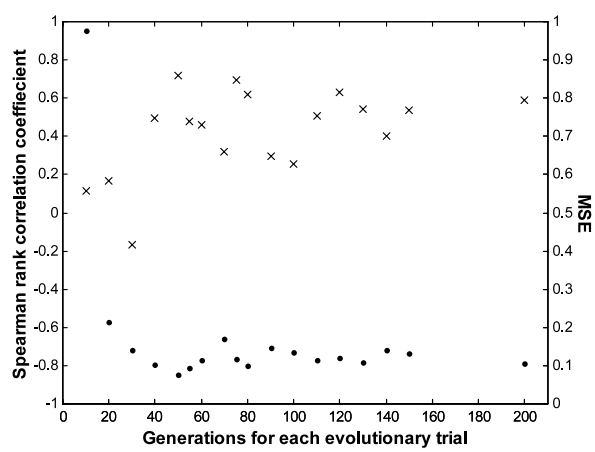

Fig. 1. The performances of evolved neural networks with 30 inputs based on Spearman rank correlation coefficient (×) and RMSE (●) given for the total number of generations in each evolutionary trial for data set 1. Performance was measured on the test set. The number of generations was varied from 10 to 200 to determine the most appropriate point to stop neural network training.

Table 1
Testing set performance as measured by RMSE and Spearman rank correlation coefficient

Number of inputs	Evolved neural networks				Hirst et al. (1994)
	Mean squared error		Spearman rank correlation coefficient		Spearman rank correlation coefficient
	3	30	3	30	27
Set 1	0.075	0.075	0.817	0.718	0.788
Set 2	0.081	0.116	0.733	0.327	0.228
Set 3	0.111	0.122	0.652	0.335	0.702
Set 4	0.097	0.172	0.773	0.443	0.838
Set 5	0.101	0.117	0.647	0.516	0.753
Mean (σ)	0.093 (0.015)	0.120 (0.034)	0.724 (0.074)	0.468 (0.161)	0.660

The best-evolved neural network with three inputs achieved a higher Spearman rank correlation coefficient (0.724) than the network with 27 inputs presented in Hirst et al. (1994) (0.660). Furthermore, comparison between the best-evolved neural networks comprising three or 30 input features in terms of RMSE favors the smaller neural network.

the remaining sets 2–5 were then trained and tested using this same architecture and parameters for 50 generations of evolution. Performance over all sets is shown in Table 1 relative to the performance offered in Hirst et al. (1994).

Some attributes carry more correlation to the known activity than others. A statistical approach was used to determine those attributes with statistically significant correlation with the known activity. Specifically, a stepwise regression was used with the values of F to enter and F to remove of 4.000 and 3.996, respectively. This analysis suggested that only one attribute for each of the three positions was well correlated to known activity. Polarizability (the response of the electron distribution to changes in the solvent or with other polar agents) was selected for positions 1 and 2, while polarity (a measure of the positive or negative charge) was selected for position 3. A new neural network architecture consisting of these three inputs, three hidden nodes and one output was used in a subsequent round of evolution. For each evolution, the number of generations was varied from 10 to 200 to determine the most appropriate point to stop neural network training. As with the previous neural network, performances were measured based on the Spearman rank correlation coefficient and the RMSE. Values for these two performance criteria are shown in Fig. 2. Based on RMSE, the most appropriate number of generations for training was again taken to be 50 for set 1. The remaining sets 2–5 were then trained and tested using this same architecture and the results are presented in Table 1.

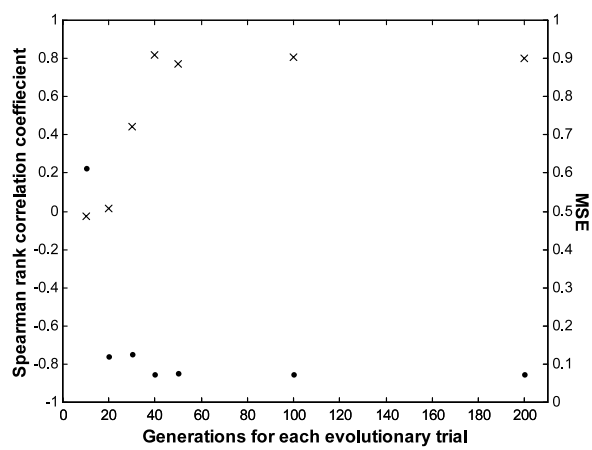

Fig. 2. The performances of evolved neural networks with three inputs based on Spearman rank correlation coefficient (×) and RMSE (●) as a function of the number of generations per evolutionary trial. Performance was measured on the test set given for the total number of generations in each evolution for data set 1. The number of generations was varied from 10 to 200 to determine the most appropriate point to stop neural network training.

3. Discussion

An assumption of all QSAR approaches is that the activity or property of a molecule is implicit in its physical structure. QSAR can then be used to develop statistical models relating known structures to activities. Given that there are many potential parameters for each molecule, neural network approaches have proven successful for activity prediction. Classical methods for training neural networks using backpropagation have been applied to QSAR problems with limited success (see above). As an alternative to backpropagation, evolutionary computation can be used as a tool to rapidly search for an appropriate set of weights that connect the nodes. Following evolution, the best-evolved neural network can then be used as a predictor for previously unknown samples in testing data. Here, we applied evolved neural networks for the inhibition of DHFR by pyrimidines as a test of the validity of this approach against a previous experiment using neural networks trained via backpropagation (Hirst et al., 1994).

To compare the performance of our neural network, we employed the Spearman rank correlation coefficient. This statistic measures the degree of linear association between two random variables. It is thought to be useful when either the data consist of ranks or when the data set is small enough to be ranked easily (Milton and Tsokos, 1983). The Spearman rank correlation coefficient was calculated for each independent run of evolution in training set 1 over a specified number of generations (10, 20, 30, …). For each of these generations, the data presented in Fig. 1 show that the Spearman rank correlation coefficient varies drastically by generation, whereas another potential performance metric, the RMSE, does not show a similar spread.

These data cast doubt on the utility of the Spearman rank correlation coefficient as a means of comparison and as a means of measuring the performance of any predictive model on these data. (These results may suggest a broader inquiry into the variability of the Spearman rank correlation coefficient more generally.) However, comparison of the evolved neural network to the data reported in Hirst et al., (1994) requires the use of the Spearman coefficient. The data in Table 1 suggest that the evolved neural network with 30 inputs, three hidden nodes, and one output node performed qualitatively worse than the network given in Hirst et al. (1994). However, the RMSE over all test sets for this same network topology was quite low (0.120), suggesting that the evolved neural network performed quite well. The variance associated with the Spearman rank correlation coefficient may due to a nonlinear association between the parameters in question.

The stepwise regression suggested that only three of the 30 available inputs might be adequate for the task at hand. Using the Spearman rank correlation coefficient as a measure of performance, the best evolved network relying on these three input features performed at a level comparable to that offered in Hirst et al. (1994). This new result should be favored in light of the maxim of parsimony, using only 10% of the previous degrees of freedom.

If attention is turned to the RMSE as a measure of performance instead of Spearman rank correlation coefficient, the best-evolved neural network with three inputs versus 30 inputs demonstrates a lower RMSE (0.093 vs. 0.120, respectively). Further study will be required to assess any statistically significant difference, but again the maxim of parsimony favors the smaller network.

It is important to note that for the research presented here, one outlier from the original data was removed (see above). This outlier was a potential source of confusion for any neural network given that it was markedly different from any other training data and would likely be a source of significant error in training and testing. It remains unclear if this outlier was a typographical error in Hirst et al. (1994), but we assume that it was used in the development of their neural network. The many noncorrelated inputs and this outlier molecule cast doubt on the validity of the performances given in Hirst et al. (1994).

It is also important to note that regardless of the measure of performance shown here, the statistical approach (relying on stepwise regression) to the physicochemical data prior to the

choice of input vectors assisted the development of a more useful predictor. Given a fixed architecture, this approach may help identify inputs that have little or no correlation with activity. However, using evolutionary computation, the topology of the neural network could be evolved simultaneously with the weights (Angeline et al., 1994). This would allow evolution to discover increasingly useful network topologies. Additional penalties could be applied for network size, giving a reward for smaller topologies of equal or better predictive accuracy.

Acknowledgements

The authors would like to thank Peter J. Angeline for assistance with software. The authors would also like to thank the reviewers for their comments, and Ray Paton for acting as editor-in-chief on this submission.

References

Altenberg, L., 1995. The schema theorem and Price's theorem. In: Whitley, L.D., Vose, M.D. (Eds.), Foundations of Genetic Algorithms 3. Morgan Kaufmann, San Mateo, CA, pp. 23–49.

Andrea, T.A., Kalayeh, H., 1991. Applications of neural networks in quantitative structure–activity relationships of dihydrofolate reductase inhibitors. J. Med. Chem. 34 (9), 2824–2836.

Angeline, P., Saunders, G., Pollack, J., 1994. Complete induction of recurrent neural networks. In: Sebald, A.V., Fogel, L.J. (Eds.), Proceedings Third Annual Conference on Evolutionary Programming. World Scientific, Singapore, pp. 1–8.

Bäck, T., Hammel, U., Schwefel, H.P., 1997. Evolutionary computation: comments on the history and current state. IEEE Trans. Evol. Comput. 1 (1), 3–17.

Efron, B., 1982. The Jackknife, the Bootstrap, and other Resampling Plans. SIAM, Philadelphia, PA.

Fogel, D.B., 1991a. System Identification through Simulated Evolution. Ginn Press, Needham, MA.

Fogel, D.B., 1991b. An information criterion for optimal neural network selection. IEEE Trans. Neural Networks 2 (5), 490–497.

Fogel, D.B., Simpson, P.K., 1993. Experiments with evolving fuzzy clusters. In: Fogel, D.B., Atmar, W. (Eds.), Proceedings of the Second Annual Conference on Evolutionary Programming. Evolutionary Programming Society, La Jolla, CA, pp. 90–97.

Fogel, D.B., Ghozeil, A., 1997. A note on representations and variation operators. IEEE Trans. Evol. Comput. 1 (2), 159–161.

Fogel, D.B., 2000. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 2nd ed. IEEE Press, Piscataway, NY.

Hirst, J.D., King, R.D., Sternberg, M.J.E., 1994. Quantitative structure–activity relationships by neural networks and inductive logic programming. I. The inhibition of dihydrofolate reductase by pyrimidines. J. Comput. Aided Mol. Des. 8, 405–420.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. Neural Networks 2, 359–366.

Kovesdi, I., Dominguez-Rodriguez, M.F., Orfi, L., Naray-Szabo, G., Varro, A., Papp, J.G., Matyus, P., 1999. Application of neural networks in structure–activity relationships. Interscience 19 (3), 249–269.

Lippmann, R.P. 1987. An introduction to computing with neural nets. IEEE ASSP Magazine, April, 4–22.

Luke, B.T., 1994. Evolutionary programming applied to the development of quantitative structure–activity relationships and quantitative structure–property relationships. J. Chem. Inf. Comput. Sci. 34, 1279–1287.

Macready, W.G., Wolpert, D.H., 1998. Bandit problems and the exploration/exploitation tradeoff. IEEE Trans. Evol. Comput. 2 (1), 2–22.

Milton, J.S., Tsokos, J.O., 1983. Statistical Methods in the Biological and Health Sciences. McGraw-Hill, Inc., New York.

Poggio, T., Girosi, F., 1990. Networks for approximation and learning. Proceedings of the IEEE 78 (9), 1481–1497.

Porto, V.W., Fogel, D.B., Fogel, L.J., 1995. Alternative neural network training methods. IEEE Expert, June, 10(3), 16–22.

Schneider, G., 2000. Neural networks are useful tools for drug design. Neural Networks 13 (1), 15–16.

So, S.-S., Helden, S.P., Van Geerstein, V.J., Karplus, M., 2000. Quantitative structure–activity relationship studies of progesterone receptor binding steroids. J. Chem. Inf. Comput. Sci. 40, 762–772.

So, S.-S., Karplus, M., 1996a. Evolutionary optimization in quantitative structure–activity relationship: an application of genetic neural networks. J. Med. Chem. 39, 1521–1530.

So, S.-S., Karplus, M., 1996b. Genetic neural networks for quantitative structure–activity relationships: improvements and application of benzodiazepine affinity for benzodiazepine/$GABA_A$ receptors. J. Med. Chem. 39, 5246–5256.

So, S.-S., Karplus, M., 1997a. Three-dimensional quantitative structure–activity relationships from molecular similarity matrices and genetic neural networks. 1. Method and validations. J. Med. Chem. 40, 4347–4359.

So, S.-S., Karplus, M., 1997b. Three-dimensional quantitative structure–activity relationships from molecular similarity matrices and genetic neural networks. 2. Applications. J. Med. Chem. 40, 4360–4371.

So, S.-S., Richards, W.G., 1992. Application of neural networks: quantitative structure–activity relationships of the derivatives of 2,4-diamino-5-(substituted-benzyl)-pyrimidines as DHFR inhibitors. J. Med. Chem. 35, 3201–3207.

Tetko, I.V., Tanchuk, V.Y., Luik, A.I., 1994. Application of an evolutionary algorithm to the structure–activity relationship. In: Sebald, A.V., Fogel, L.J. (Eds.), Proceedings of the Third Annual Conference on Evolution Programming. World Scientific, New Jersey, 1994, pp. 109–118.

Vose, M.D., Wright, A.H., 2001. Form invariance and implicit parallelism. Evol. Comput. 9 (3), 355–370.

Neill, S.R.St.J., 1974. Experiments on whether schooling by their prey affects the hunting behavior of cephalopods and fish predators. J. Zool. Lond., 172, 549–569.