# Evolving Artificial Neural Network Structures: Experimental Results for Biologically-Inspired Adaptive Mutations

Damon A. Miller
Dept. of Electrical and Computer Engineering
Western Michigan University
Kalamazoo, MI 49008
Email: d.a.miller@ieee.org

Rodrigo Arguello
Dept. of Electrical and Computer Engineering
Western Michigan University
Kalamazoo, MI 49008
Email: rodrigo.arguello@stryker.com

Garrison W. Greenwood
Dept. of Electrical and Computer Engineering
Portland State University
Portland, OR 97207
Email: greenwood@ieee.org

***Abstract*— Previous work has suggested incorporating a biologically inspired mutation operator that reflects vertebrate neuron growth and death rates into an evolutionary algorithm for evolving artificial neural network structures. This paper further investigates this proposed approach by presenting experimental results for two classifier problems.**

## I. Introduction

The problem of how to select a multilayer feed-forward neural network (MFNN) structure for a particular problem remains an important but unresolved issue. To date no approach is able to provide a definitive solution to this problem, although a variety of methods are available [1], [2]. Bayesian approaches are based on a probability distribution of possible weight solutions [3]. Finding a single MFNN structure suitable for hardware implementation remains something of an art.

One of the most desirable characteristics of a properly sized MFNN is its ability to provide correct responses to novel inputs, i.e. input patterns not found in the training set. This capability is often referred to as an ability of the MFNN to "generalize" [4]. Generalization is actually an ill-posed problem, since any approximating function cannot in general provide an error free approximation to a continuous input-output mapping based on a finite number of training points. The general rule of thumb is to use the least complex MFNN that provides good performance over one or more input-output mapping examples.

Evolutionary computation (EC) can be used to train a MFNN (i.e. determine an acceptable set of network weights), find a suitable MFNN architecture, or both [5]. In contrast to existing methods to evolve MFNNs, the EC method examined in this paper utilizes a biologically inspired mutation mechanism which encourages either the addition or removal of neurons as the MFNN population is evolved. Reference [6] presented preliminary experimental results for this strategy; this paper presents additional empirical evidence to illustrate the power of our biologically-inspired adaption method.

## II. Methodology

The biologically inspired neuron mutation algorithm (Fig. 1) is based on the $(\mu + \mu)$ evolutionary strategy [7], [8]. Each member of a population of $\mu$ parents is mutated to produce a single offspring. This population is trained for a prescribed number of standard error back-propagation (EBP) [9], [10] training epochs. The $\mu$ most fit individuals are selected based on their performance on a fitness data set. These individuals comprise the initial population for the next generation. The next section (Experimental Results) will provide relevant parameter settings, e.g. population sizes and learning rates, as these vary from case to case.

The inspiration for the proposed mutation operator is the observation that the number of synapses and neurons in vertebrate brains follow a consistent trend. Synaptic connections are rapidly produced during initial stages of brain development and a great number of these connections are subsequently reduced or pruned. Neuron growth and death follow a similar predictable pattern. It is thought that more important synapses and neurons are favored for survival. For biological details and references, see [6].

The properties of the proposed mutation operator were selected to mimic the described biological synaptic and neural growth and death trends. The mutation operator potentially adds or deletes neurons in a given layer; the probability of adding or deleting neurons is directly proportional to the difference between the current number of neurons in that layer and a tent shaped graph that specifies a target number of neurons for each generation. This tends to favor more neurons in earlier generations and less neurons in later generations (Fig. 2).

## III. Experimental Results

### A. Preliminary Notes

The MFNNs considered are feedforward types with full connectivity between adjacent layers only. Neurons have a

0-7803-8515-2/04/$20.00 ©2004 IEEE

## Biologically Inspired Neuron Mutation Algorithm

**Purpose:** Find a suitable MFNN architecture and associated weights for a given training problem

1. SELECT number of MFNN layers $I$ and the neuron growth and death parameters for each layer as follows (layer $I = 1$ is the input layer). First, choose

| | |
|---|---|
| $N0_i$ | initial number of neurons in layer $i$ |
| $Np_i$ | peak number of neurons in layer $i$ |
| $\gamma p_i$ | time index where $Np_i$ occurs |
| $Ne_i$ | end number of neurons in layer $i$ |
| $\gamma e_i$ | ending index where $Ne_i$ occurs |

and compute

$$\beta_{i0} = N0_i \qquad \beta_{i2} = (Np_i - Ne_i)/(\gamma p_i - \gamma e_i) \tag{1}$$

$$\beta_{i1} = (Np_i - N0_i)/\gamma p_i \qquad \beta_{i3} = Np_i - \beta_{i2} * \gamma p_i. \tag{2}$$

The resulting piecewise linear tent-shaped neuron schedule for generation $t$ is

$$\sigma_i^{t+1} = \begin{cases} \lfloor (\beta_{i1} t + \beta_{i0} + 0.001) \rfloor & 0 \geq t \leq \gamma p_i \\ \lfloor (\beta_{i2} t + \beta_{i3} + 0.001) \rfloor & \gamma p_i < t \leq \gamma e_i. \end{cases} \tag{3}$$

2. CREATE and INITIALIZE $P = \{p_1, p_2, \ldots, p_\mu\}$ parent MFNNs with size $N0_1 - N0_2 - \cdots - N0_I$. Weights are initialized using a zero mean normal distribution with variance $1/d$, where $d$ is the neuron fan-in (including the bias) [3].
3. FOR EACH GENERATION:
   a. TRAIN parents $P$ for $T$ training epochs using the EBP algorithm with learning rate $\eta$ using the training set. $T$ is user selectable.
   b. CREATE A COPY (offspring) of each parent and save in $O = \{o_1, o_2, \ldots, o_\mu\}$.
   c. MUTATE the offspring $O$ according to the biologically inspired neuron growth schedule:
      i. Choose a layer $i$ to mutate with probability $p_i$.
      ii. Compute $diff_i = \hat{\sigma}_i^t - \sigma_i^t$, where $\hat{\sigma}_i^t$ is the number of neurons (excluding the bias) in offspring neuron layer $i$.
      iii. If $diff_i < 0$, choose a natural number $n$ with uniform probability from the interval $[1, diff_i]$; add $n$ neurons to layer $i$ and initialize the new neuron weights as described above.
      iv. If $diff_i > 0$, choose a natural number $n$ with uniform probability from the interval $[1, diff_i]$; randomly choose and then delete $n$ neurons in layer $i$. At least one neuron must remain in the layer.
   d. TRAIN the offspring $O$ for $T$ training epochs using the EBP algorithm with learning rate $\eta$.
   e. Let the population $W = P \cup O$.
   f. Determine the POPULATION FITNESS using the fitness set.
   g. Update as needed $MFNN^*$ which is the MOST FIT NETWORK found to date.
   h. SAVE the $\mu$ most fit members of the population in $P$.
4. OUTPUT $P$ and $MFNN^*$

Fig. 1. Proposed algorithm

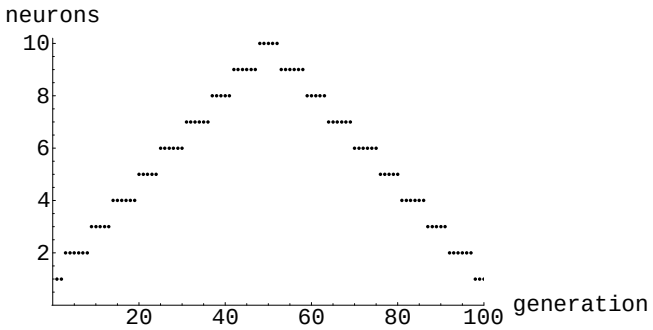


Fig. 2. Example neuron growth and death schedule. This curve directs the mutation operator to prefer an increasing number of neurons in beginning generations and a decreasing number of neurons in later generations. The "staircase" appearance is a consequence of a floor operator.

sigmoidal activation function

$$f(net) = \frac{1}{1 + e^{-net}} \tag{4}$$

where $net$ is the scalar product of the neuron input and neuron weight vectors [4]. In this paper MFNN classifiers use a 1-of-$M$ encoding, i.e. the MFNN has $M$ outputs for a classification problem with $M$ classes. The largest MFNN output indicates the network classification for an input pattern. Note that in this paper the following notation is used to identify a MFNN structure: (number of inputs less the bias input)–(number of neurons in first hidden layer less the bias unit)–(number of neurons in second hidden layer less the bias unit, if any)–(number of outputs). Thus for a 8-7-12-2 MFNN there are eight inputs plus a bias input, the first layer has seven neurons plus a bias input for the second hidden layer, the second hidden layer has twelve neurons plus a bias input for the output layer, and the output layer has two neurons.

### *B. Gaussian Classifier (GC)*

This section applies the proposed methodology to a classifier problem as described in [1]. Samples are drawn from two significantly overlapping Gaussian distributions with means $\mu$ and covariance matrices $\mathbf{\Sigma}$ given as

$$\mu_1 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T \qquad \mathbf{\Sigma}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5}$$

and

$$\mu_2 = \begin{bmatrix} 2 & 0 \end{bmatrix}^T \qquad \mathbf{\Sigma}_2 = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}. \tag{6}$$

This example was chosen since (i) the readily obtained ideal Bayesian classifier provides a basis of comparison for the MFNN performance, (ii) Haykin has provided architecture selection results for this problem, and finally (iii) this two dimensional problem is easily visualized. Haykin used a training set, fitness set, and test set containing 1,000, 32,000, and 32,000 samples, respectively. Using a test set size of 32,000 provides a 99 percent likelihood estimate of the actual MFNN classifier performance [1].

The parameters of Table I were used to generate the results of Table II. The number of samples in the training set, fitness set, and test set match Haykin. Haykin determined that a 2-2-2 MFNN is adequate for this problem; an ensemble of 20 such networks provided a mean classifier error of 20.3% for 32,000 test vectors [1]. This classifier error performance is considered to be the "solution" although this "solution" is not optimal; the intent is simply to provide an example solution as a basis for comparison. Note that the two layer MFNN discovered in the GC 4 run provided a 18.48% classifier error on the test set, which is on par with the optimal Bayesian classifier error of ≈18.49% [1]. The discovered MFNN structures certainly cannot be considered parsimonious; however, the algorithm evolved structures that provided acceptable performance levels. Representative plots are provided in Fig. 3.

### *C. Diabetes Classifier (DB2)*

This classification problem is taken from the PROBEN1 set of neural network benchmark problems [11]. This data set (diabetes2.dt) contains examples of individuals that have been diagnosed either as diabetic or not diabetic. The input to the classifier consists of eight measurements of personal data (e.g. age) and other medical data (e.g. blood pressure). Experimental parameters and results are provided in Tables I and II. The "solution" classifier performance is taken from [11, Table 5] and represents the performance of the indicated **fully connected (includes shortcut connections)** 8-16-8-2 MFNN architecture that used linear output neurons. This architecture was selected after testing several alternative structures. The resulting "solution" classifier performance values are not advertised to be optimal but are included to provide a basis of comparison. The 8-16-8-2 architecture without shortcut connections provided a 26.42% classification error [11, Table 5]. Note that the proposed EC algorithm was constrained to have no shortcut connections and to use only sigmoidal neurons. The algorithm discovered MFNNs with comparable performance.

## IV. OBSERVATIONS AND CONCLUSIONS

Fig. 4 provides an interesting set of curves for three of the diabetes classifier test cases. The DB2 3 case provided the worst classifier performance and the most "static" change in the number of neurons in each layer. The DB2 6 case provided the most "dynamic" adaptation behavior and the best classifier performance. The DB2 1 case is intermediate in both adaptation behavior and classifier performance. A general but inconsistent trend of runs with less "dynamic" adaptation curves providing worse classifier performance was noted for some of the Gaussian classifier test cases as well. The best performing decision boundary was provided by a MFNN discovered with a relatively "dynamic" neuron adaptation curve (GC 4). Few surviving offspring result in "static" adaptation curves; this suggests the possibility of using the offspring survival rate as a feedback mechanism for adjusting algorithm parameters, e.g. the length of EBP training. These parameters might also be candidates for adaptive selection.

The results presented in this paper are promising and suggest that the biologically inspired neuron mutation algorithm [6] may be an effective strategy for simultaneously determining a suitable MFNN structure and its weights. More rigorous testing is required; ideally, this would include verification of and comparison to the results of [11]. Adaptation at the connection level as opposed to the neuron level as suggested in [6] also needs to be considered.

## REFERENCES


[1] S. Haykin, *Neural Networks*. Upper Saddle River, New Jersey: Prentice-Hall, 2nd ed., 1994.

[2] R. Reed, "Pruning algorithms—a survey," *IEEE Transactions on Neural Networks*, vol. 4, pp. 740–747, September 1993.

[3] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York: Oxford University Press Inc., 1995.

[4] J. M. Zurada, *Introduction to Artificial Neural Systems*. Boston, Massachusetts: PWS Publishing Company, 1992.

TABLE I

ALGORITHM PARAMETERS CORRESPONDING TO THE RESULTS OF TABLE II

| learning problem | $\mu$ | $I$ | $N0_1$ | $Np_1$ | $Ne_1$ | $\gamma p_1$ | $\gamma e_1$ | $p_1$ | $N0_2$ | $Np_2$ | $Ne_2$ | $\gamma p_2$ | $\gamma e_2$ | $p_2$ | $T$ | $\eta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GC 1 | 20 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 10 | 0.01 |
| GC 2 | 20 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 1 | 0.01 |
| GC 3 | 20 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 5 | 0.01 |
| GC 4 | 20 | 2 | 1 | 10 | 1 | 50 | 100 | 0.75 | 1 | 10 | 1 | 50 | 100 | 0.25 | 5 | 0.01 |
| GC 5 | 20 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 20 | 0.01 |
| GC 6 | 20 | 1 | 1 | 30 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 40 | 0.01 |
| GC 7 | 40 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 10 | 0.01 |
| GC 8 | 80 | 1 | 1 | 10 | 1 | 50 | 100 | 0.75 | - | - | - | - | - | - | 10 | 0.01 |
| GC 9 | 100 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 10 | 0.01 |
| GC 10 | 40 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 5 | 0.01 |
| GC 11 | 80 | 2 | 1 | 10 | 1 | 50 | 100 | 0.75 | 1 | 10 | 1 | 50 | 100 | 0.25 | 10 | 0.01 |
| GC 12 | 20 | 2 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 40 | 0.01 |
| GC 13 | 20 | 1 | 1 | 10 | 1 | 50 | 100 | 0.75 | 1 | 10 | 1 | 50 | 100 | 0.25 | 40 | 0.01 |
| GC 14 | 10 | 1 | 1 | 10 | 1 | 50 | 100 | 1.0 | - | - | - | - | - | - | 5 | 0.01 |
| GC 15 | 80 | 1 | 1 | 10 | 1 | 50 | 100 | 0.75 | 1 | 10 | 1 | 50 | 100 | 0.25 | 5 | 0.01 |
| GC 16 | 20 | 1 | 1 | 10 | 1 | 50 | 100 | 0.75 | 1 | 10 | 1 | 50 | 100 | 0.25 | 80 | 0.01 |
| DB2 1 | 80 | 2 | 1 | 30 | 1 | 250 | 500 | 0.75 | 1 | 30 | 1 | 250 | 500 | 0.25 | 2 | 0.008 |
| DB2 2 | 20 | 2 | 5 | 20 | 5 | 100 | 200 | 0.75 | 2 | 10 | 5 | 100 | 200 | 0.25 | 2 | 0.008 |
| DB2 3 | 20 | 2 | 5 | 20 | 5 | 100 | 200 | 0.75 | 2 | 10 | 5 | 100 | 200 | 0.25 | 1 | 0.008 |
| DB2 4 | 20 | 2 | 1 | 30 | 1 | 250 | 500 | 0.75 | 1 | 30 | 1 | 250 | 500 | 0.25 | 1 | 0.008 |
| DB2 5 | 20 | 2 | 5 | 20 | 5 | 100 | 200 | 0.75 | 2 | 10 | 5 | 100 | 200 | 0.25 | 1 | 0.008 |
| DB2 6 | 40 | 2 | 1 | 30 | 1 | 250 | 500 | 0.75 | 1 | 30 | 1 | 250 | 500 | 0.25 | 1 | 0.008 |

TABLE II

EXPERIMENTAL RESULTS FOR THE BIOLOGICALLY INSPIRED NEURON MUTATION ALGORITHM

| learning problem | classification performance % error | | | | | architecture | | |
|---|---|---|---|---|---|---|---|---|
| | training set | fitness set | | test set | | results | | |
| | algorithm | algorithm | "solution" | algorithm | "solution" | algorithm | "solution" | initial |
| GC 1 | 17.60% | 18.22% | 20.3% | 18.80% | 20.3% | 2-6-2 | 2-2-2 | 2-1-2 |
| GC 2 | 17.90% | 18.75% | 20.3% | 19.23% | 20.3% | 2-7-2 | 2-2-2 | 2-1-2 |
| GC 3 | 17.80% | 18.35% | 20.3% | 18.74% | 20.3% | 2-10-2 | 2-2-2 | 2-1-2 |
| GC 4 | 17.60% | 18.13% | 20.3% | 18.48% | 20.3% | 2-9-9-2 | 2-2-2 | 2-1-1-2 |
| GC 5 | 17.40% | 18.24% | 20.3% | 18.64% | 20.3% | 2-7-2 | 2-2-2 | 2-1-2 |
| GC 6 | 16.80% | 18.14% | 20.3% | 18.62% | 20.3% | 2-25-2 | 2-2-2 | 2-1-2 |
| GC 7 | 17.80% | 18.32% | 20.3% | 18.69% | 20.3% | 2-9-2 | 2-2-2 | 2-1-2 |
| GC 8 | 17.40% | 18.25% | 20.3% | 18.72% | 20.3% | 2-5-2 | 2-2-2 | 2-1-2 |
| GC 9 | 17.80% | 18.23% | 20.3% | 18.82% | 20.3% | 2-6-2 | 2-2-2 | 2-1-2 |
| GC 10 | 18.00% | 18.31% | 20.3% | 18.73% | 20.3% | 2-7-2 | 2-2-2 | 2-1-2 |
| GC 11 | 17.60% | 18.17% | 20.3% | 18.55% | 20.3% | 2-9-9-2 | 2-2-2 | 2-1-1-2 |
| GC 12 | 17.60% | 18.16% | 20.3% | 18.66% | 20.3% | 2-3-6-2 | 2-2-2 | 2-1-2 |
| GC 13 | 17.70% | 18.28% | 20.3% | 18.68% | 20.3% | 2-9-2 | 2-2-2 | 2-1-1-2 |
| GC 14 | 17.70% | 18.25% | 20.3% | 18.66% | 20.3% | 2-6-2 | 2-2-2 | 2-1-2 |
| GC 15 | 17.60% | 18.36% | 20.3% | 18.78% | 20.3% | 2-10-2 | 2-2-2 | 2-1-2 |
| GC 16 | 17.50% | 18.19% | 20.3% | 18.53% | 20.3% | 2-4-2 | 2-2-2 | 2-1-2 |
| DB2 1 | 21.88% | 21.35% | 21.35% | 25.00% | 23.44% | 8-25-25-2 | 8-16-8-2 | 8-1-1-2 |
| DB2 2 | 21.09% | 21.35% | 21.35% | 27.08% | 23.44% | 8-6-6-2 | 8-16-8-2 | 8-5-2-2 |
| DB2 3 | 33.85% | 34.38% | 21.35% | 26.56% | 23.44% | 8-5-7-2 | 8-16-8-2 | 8-5-2-2 |
| DB2 4 | 22.40% | 22.40% | 21.35% | 26.56% | 23.44% | 8-6-13-2 | 8-16-8-2 | 8-1-1-2 |
| DB2 5 | 18.75% | 19.79% | 21.35% | 25.52% | 23.44% | 8-19-19-2 | 8-16-8-2 | 8-5-2-2 |
| DB2 6 | 20.31% | 21.09% | 21.35% | 22.40% | 23.44% | 8-7-12-2 | 8-16-8-2 | 8-1-1-2 |

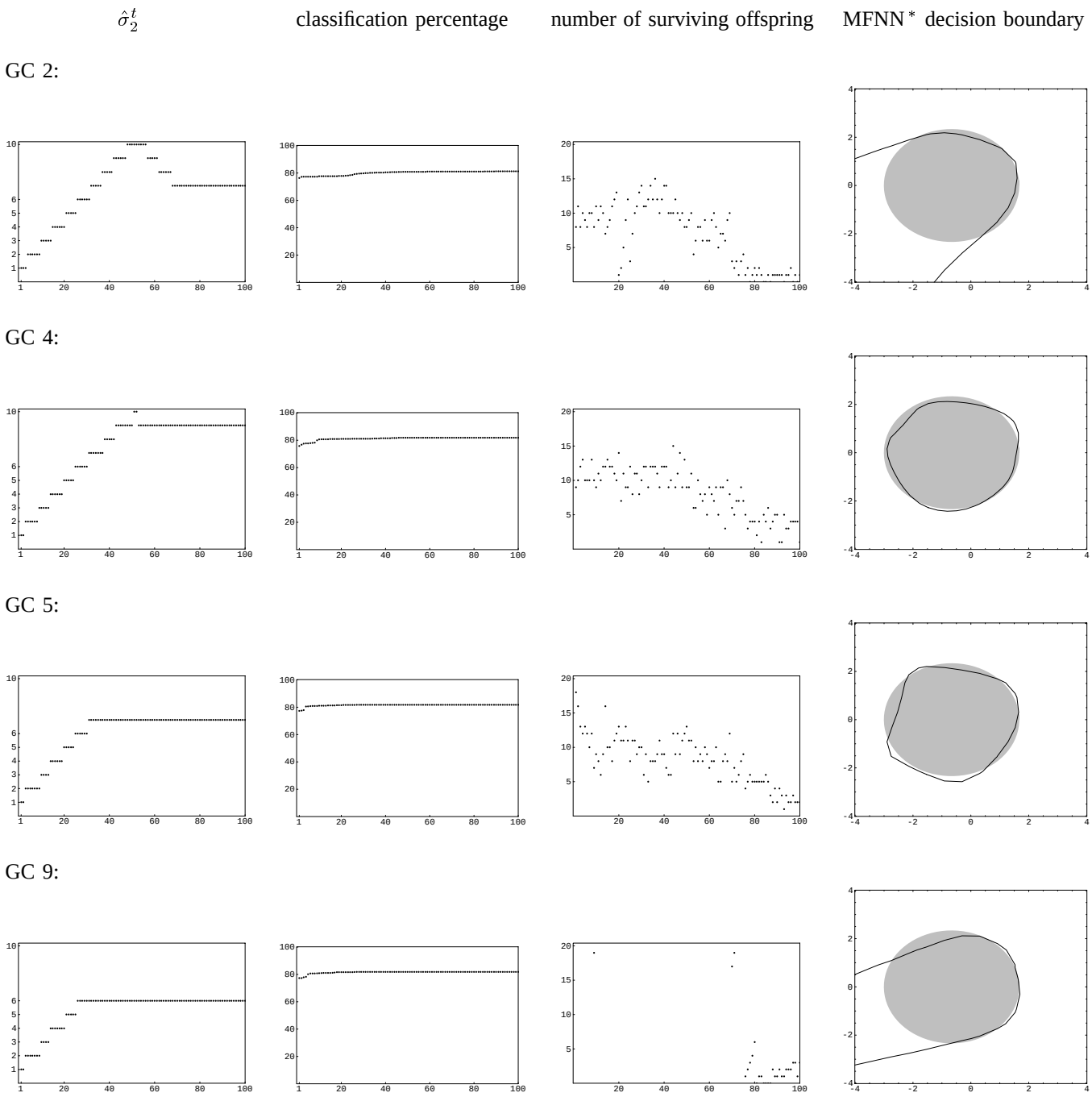


Fig. 3. Plots for four Gaussian classifier cases. Horizontal axes in first three columns are the generation number. Last column axes are in the pattern space. Variable $\hat{\sigma}_1^t$ is the number of neurons in the hidden layer of the best MFNN to date and the classification percentage is of the best MFNN to date. The decision boundary is that provided by the best MFNN discovered (MFNN*). The ideal Bayesian classifier decision region for class 1 is the shaded gray circle.

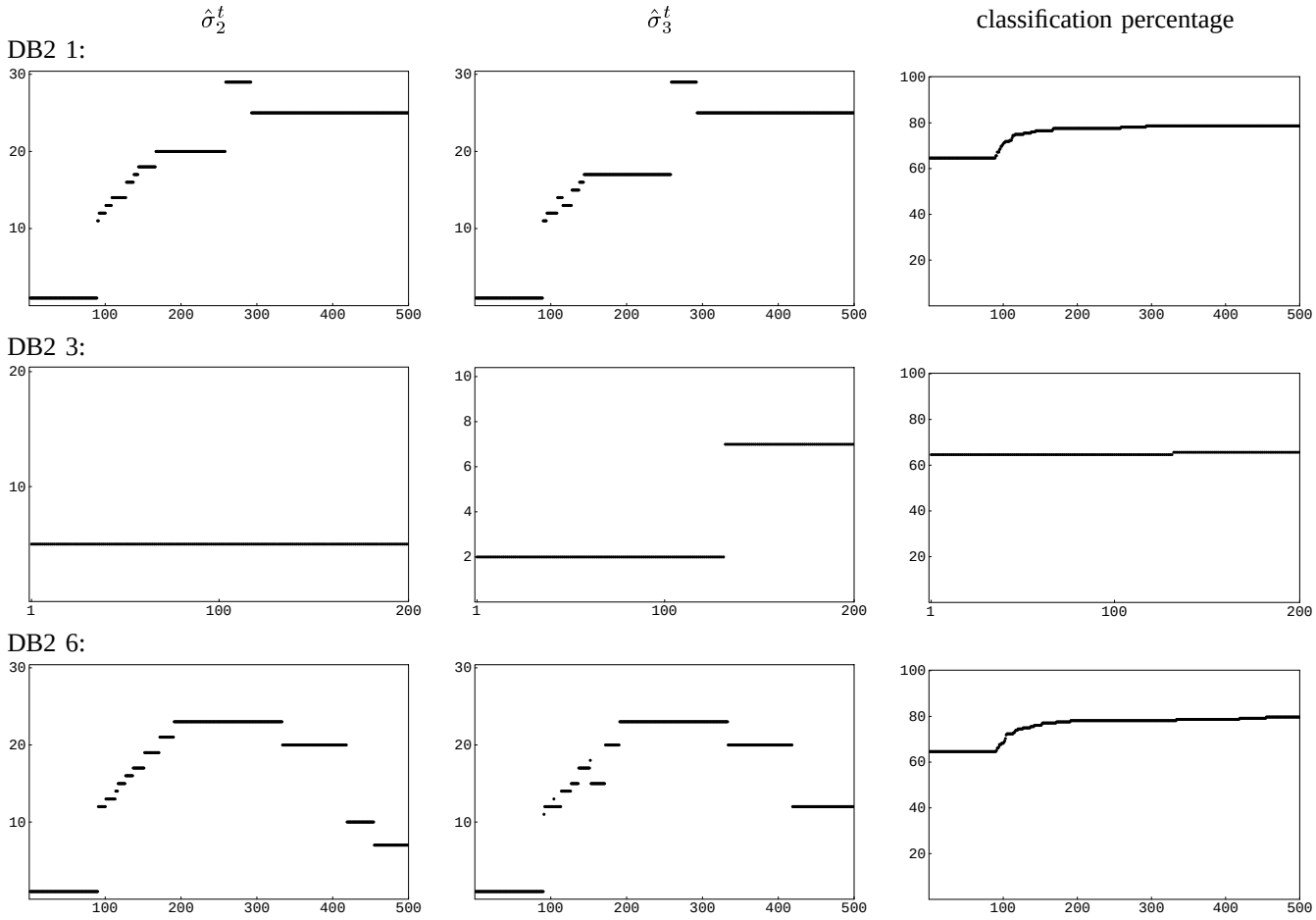


Fig. 4. Plots for several DB2 test cases. Variables $\hat{\sigma}_1^t$ and $\hat{\sigma}_2^t$ are the number of neurons in the corresponding hidden layer of the best MFNN to date. The last column is the classification percentage of the best MFNN to date. Horizontal axes are the generation number.


[5] E. Vonk, L. C. Jain, and R. P. Johnson, *Automatic generation of neural network architecture using evolutionary computation*, vol. 14 of *Advances in Fuzzy Systems—Applications and Theory*. Singapore: World Scientific, 1997.

[6] D. A. Miller, G. Greenwood, and C. Ide, "On the use of biologically-inspired adaptive mutations to evolve artificial neural network structures," in *Proc. of the First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, (San Antonio, TX), pp. 24–32, May 11–12 2000.

[7] I. Rechenberg, "Cybernetic solution path of an experimental problem," *Royal Aircraft Establishment*, 1965. Library Translation No. 1122.

[8] H. P. Schwefel, *Kybernetische evolution als strategie der experimentellen fors chung in der stromungstechnik*. PhD thesis, Tech. Univ. of Berlin, 1965.

[9] P. J. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA, 1974.

[10] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by back-propagating errors," in *Parallel Distibuted Processing: Explorations in the Microstructure of Cognition* (D. E. Rumelhart and J. L. McClelland, eds.), vol. 1, ch. 8, Cambridge, MA: MIT Press, 1986.

[11] L. Prechelt, "PROBEN1 — A Set of Neural Network Benchmark Problems and Benchmarking Rules," tech. rep., Universität Karlsruhe, Karlsruhe, Germany, September 1994. Available at http://wwwipd.ira.uka.de/~prechelt/.