Available online at www.sciencedirect.com
ScienceDirect
Knowledge-Based Systems 21 (2008) 217–224
Knowledge-Based SYSTEMS
www.elsevier.com/locate/knosys

Evolving a dynamic predictive coding mechanism for novelty detection

Simon J. Haggett [a,*], Dominique F. Chu [a], Ian W. Marshall [b]

[a] Computing Laboratory, University of Kent, Canterbury CT2 7NF, UK
[b] Lancaster Environment Centre, Lancaster University, Lancaster LA1 4YQ, UK

Available online 23 November 2007

Abstract

Novelty detection is a machine learning technique which identifies new or unknown information in data sets. We present our current work on the construction of a new novelty detector based on a dynamical version of predictive coding. We compare three evolutionary algorithms, a simple genetic algorithm, NEAT and FS-NEAT, for the task of optimising the structure of an illustrative dynamic predictive coding neural network to improve its performance over stimuli from a number of artificially generated visual environments. We find that NEAT performs more reliably than the other two algorithms in this task and evolves the network with the highest fitness. However, both NEAT and FS-NEAT fail to evolve a network with a significantly higher fitness than the best network evolved by the simple genetic algorithm. The best network evolved demonstrates a more consistent performance over a broader range of inputs than the original network. We also examine the robustness of this network to noise and find that it handles low levels reasonably well, but is outperformed by the illustrative network when the level of noise is increased.

Keywords: Novelty detection; Neural networks; Neuroevolution; Evolutionary algorithms

1. Introduction

A novelty detector is a machine learning system that identifies new or unknown data that it was not aware of during its training phase [4]. Novelty detection is important in practical applications where large data sets are being processed. In a sensor array which continuously monitors an unpredictable environment, a constant stream of data is produced by each sensor. A large proportion of this data will be redundant since it describes information already known. The crucial information is that which indicates that change has occurred, since this may require some action to take place. Novelty detection can be used in this case to highlight such data.

Because novelty detection is an extremely challenging task, there are currently a number of different approaches [3]. A comprehensive survey of approaches using neural networks is given by Markou and Singh [4]. Marsland et al. [5] propose a novelty detector which employs a Habituating Self Organising Map (HSOM). An input layer is connected to a clustering layer which represents the feature space. Each input vector is classified by associating it with a neuron in the clustering layer as follows. The neuron in the clustering layer with the smallest distance to the input vector (where distance is defined by the sum of the squared difference between each element of the input vector and the value held in the given node in the clustering layer) fires for that input vector. Each node in the clustering layer is connected to the output neuron via a habituable synapse, which not only weakens (habituates) when the corresponding node in the clustering layer fires frequently, but also strengthens (dishabituates) when the node fires infrequently. The value of the output neuron for a given input vector indicates the novelty of that vector. Habituation enables the network to learn on-line and cope with changing environments. However, since the size of the network remains fixed one limitation is that on-line learning can cause saturation in the network. This is when all synapses

* Corresponding author.
E-mail addresses: simon.haggett@acm.org (S.J. Haggett), D.F.Chu@kent.ac.uk (D.F. Chu), I.W.Marshall@lancaster.ac.uk (I.W. Marshall).

0950-7051/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.knosys.2007.11.007

habituate, resulting in novel input vectors being misclassified as normal [6]. Marsland et al. [6,7] extend this work by proposing the 'Grow When Required' (GWR) network as an improvement to the HSOM. The GWR network is a new type of clustering map that allows new nodes to be created as required, thus overcoming the problem of saturation seen in the HSOM.

Predictive coding is a technique used in areas such as image and speech compression [2]. In image compression, this technique attempts to predict the value of a given pixel based on the values of neighbouring pixels. A difference signal, holding the difference between the predicted and observed values of the given pixel, is then used to represent that pixel. When the predicted value is close to the observed value, this difference signal will have a smaller magnitude, which means that it can be represented more compactly. In turn, this allows the image as a whole to be represented in a compressed form. If we assume that novel values are likely to be unpredictable, then the difference signal can be used to determine the novelty of the observed value.

Hosoya et al. [1] propose a possible neural network model of circuits in the retina. This model performs a dynamical version of predictive coding in that it adapts on-line to the changing visual scene. As animals move through their environment, they tend to encounter visual scenes which differ strongly in their statistical properties. For example, the scene in a woodland environment is likely to have strong correlation between vertically separated points and weak correlation between horizontally separated points, whilst the scene in a sandy environment, such as a desert or beach, is likely to have correlation between points only in small localities. By adapting to the changing image statistics, a predictive coder is able to maintain its efficiency as the visual scene changes.

We wish to develop a new novelty detector which is based on dynamic predictive coding. This form of predictive coding is a promising model to base a novelty detector on because of its capability to learn on-line the current norm conditions and adapt to changes in these conditions over time. Unlike Marslands GWR [6,7], which is also capable of on-line learning, this method of novelty detection will learn the statistical relationships between values and report novelty when those relationships change. Inputs are classified depending on the statistical relationships between their elements, as opposed to Euclidean distance to a feature prototype. Therefore, inputs which are represented by multiple classes in GWR may be represented by a single class in our proposed approach.

In this paper, we present our current work on the construction of a new novelty detector based on dynamic predictive coding. We compare three evolutionary algorithms for the task of evolving a neural network structure to give an optimal performance over stimuli from a number of artificially generated visual environments. We also examine the robustness of the evolved structure when noise is introduced to its inputs. The remainder of this paper is organised as follows. In Section 2, we describe a neural network capable of dynamic predictive coding and identify a limitation of this network. Section 3 describes three evolutionary algorithms used to search for a neural network structure which gives an optimal performance according to two basic criteria. In Section 4, we present experimental results from using these algorithms and examine the robustness of the best evolved network to random noise. Section 5 discusses these results and the comparative performance of the evolutionary algorithms. Finally, Section 6 concludes this paper and briefly outlines how we plan to continue this work.

2. Novelty detection using dynamic predictive coding

The neural network model proposed by Hosoya et al. [1] is a feedforward network which represents a neural circuit found in the retina [11]. In this model, input neurons connect to output neurons via fixed-weight synapses and/or modifiable synapses. The fixed-weight synapse from input x_j to output y_i is represented as b_{ij}, and the modifiable synapse between these neurons is represented as a_{ij}. The output y_i of the ith output neuron is given by the following equation:

$$y_i = \sum_j (b_{ij} + a_{ij}) x_j \tag{1}$$

At each output neuron, the network attempts to predict the sum of the inputs received through fixed synapses and subtract this prediction from the neurons input. The modifiable synapse weights are modulated according to the anti-hebbian learning rule shown in Eq. 2 to form this prediction

$$\frac{da_{ij}}{dt} = \frac{-a_{ij} - \beta\langle y_i x_j\rangle}{\tau} \quad \tau,\ \beta > 0 \tag{2}$$

$\langle y_i x_j\rangle$ is the time-averaged correlation between input x_j and output y_i and is sampled over m previous time steps up to the current time step t

$$\langle y_i x_j\rangle = \frac{1}{m} \sum_{k=t-m+1}^{t} y_i(k) x_j(k) \tag{3}$$

where $y_i(k)$ and $x_j(k)$ are the values of y_i and x_j, respectively, at time k. This anti-hebbian learning rule causes the modifiable synapses to weaken when the activity at the presynaptic and postsynaptic neurons is correlated and strengthen when the activity is anti-correlated. The parameters β and τ control the networks sensitivity to the correlation signal and the rates of learning and decay, respectively.

To illustrate this model, the following single layer example neural network was given by Hosoya et al. [1]. Consider a 4×4 pixel greyscale image where each pixel provides a single input to the network. The network has a single output neuron y which aims to predict the sum of the centre 2×2 pixels given the correlational relationships between those pixels and the neighbouring 'surround' pixels. All

pixels are connected to the output neuron by modifiable synapses. In addition, the 2×2 centre pixels are also connected to the output neuron via fixed-weight synapses with weight 1. In the networks initial state, each modifiable synapse has a weight of 0, meaning that the output of the network is initially the sum of the centre 2×2 pixels. Over time, the modifiable synapses have their weights updated by the anti-hebbian learning rule (Eq. (2)) such that a prediction of the sum of the 2×2 pixels is formed and subtracted from the observed sum.

Hosoya et al. [1] demonstrate the operation of this illustrative neural network over a number of artificially generated visual 'environments'. Four of these environments were flickering greyscale images with perfect correlational relationships and one of these environments, titled ''random'', was a flickering environment with no correlation between pixels. The four environments with perfect correlational relationships were a flickering uniform field, a flickering checkerboard pattern and flickering vertical and horizontal bars. Each pixel was updated every u timesteps with an independently drawn value from a standard normal distribution. Finally, a 'none' environment was used which was defined as a steady grey screen. Fig. 1 illustrates the environments. To analyse the performance of the network, Hosoya et al. [1] observed how the sensitivity of the output neuron to the uniform, checkerboard, vertical bar and horizontal bar environments varied during the simulation. Sensitivity of the output neuron y to a given environment E is defined by Hosoya et al. [1] as the square root of the averaged variance of y, taken over stimuli from environment E (Eq. (5) in Section 3).

We use this network as the basis for a novelty detector utilising dynamic predictive coding. To improve our understanding and identify any limitations, we constructed a simulation in which we observed how this network responded when shown a series of visual environments. In this simulation, each environment was shown for 2500 time steps. After every time step, the weights of the network were frozen and the variance of the output neuron $\mathrm{var}(y)$ sampled for stimuli from each environment. After every 100 time steps, these variances were averaged and the sensitivity to each environment calculated. We used a value of 0.4 for β and a value of 500 for τ, both determined experimentally. The parameter u was set to 1 to give maximum flicker.

We also introduced two new diagonal environments (shown in Fig. 2) and observed how the network responded to these. As with the existing environments, each diagonal environment was shown to the network for 2500 time steps.

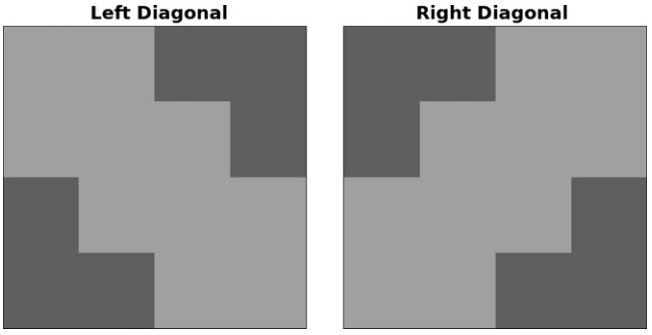

Fig. 2. The new diagonal environments.

To verify our implementation of this example network, we first tested it with the original environments used in [1]. Fig. 3 illustrates how the networks sensitivity to the uniform environment varies during the time course of the simulation. For this network, sensitivity is scaled such that a sensitivity of 1 is defined by the sensitivity of the network to environments in its unadapted state. The network was shown each environment in the order implied by the horizontal axis.

When the network was shown the ''random'' environment, its sensitivity to the uniform environment fell slightly. This behaviour was observed for all environments monitored and is also demonstrated in [1]. When the network adapted to the uniform environment, its sensitivity to that environment fell considerably. In this state, the network considers the uniform environment to be known and therefore not novel. When the network was subsequently shown the checkerboard environment, its sensitivity to the uniform environment recovered. The network forgets about the uniform environment and classifies it as novel again. Fig. 4 illustrates how the networks sensitivity to the original environments varied through this simulation.

We also performed a simulation using the diagonal environments. Fig. 5 shows how the networks sensitivity varied through the simulation. As the network adapts to the uniform, checkerboard, vertical bar and horizontal bar environments, its sensitivity to the diagonal environments falls to approximately 0.75. As the network adapts to one diagonal environment, the sensitivity to the other diagonal environment rises above 1. Here, the output of the network is greater than the sum of the centre 2×2 patch. Since the goal of predictive coding is to compress the observed value, this behaviour is undesired.

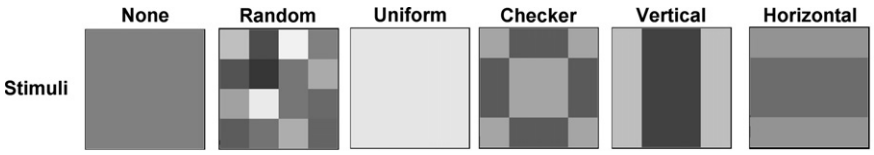

Fig. 1. The artificial environments specified in [1].

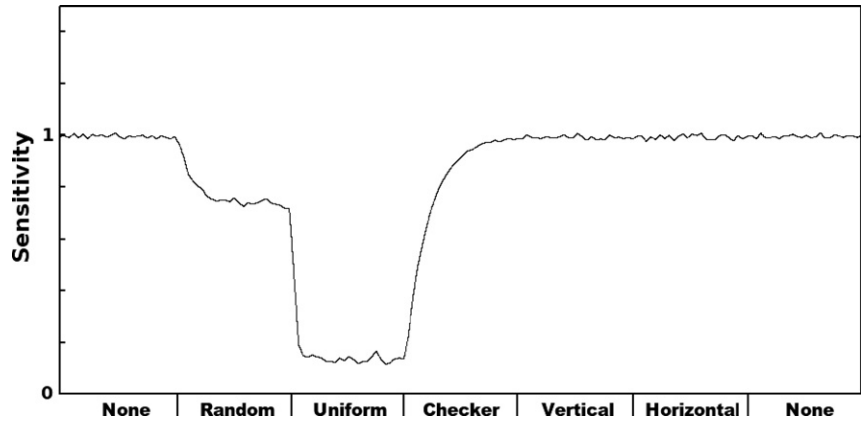

Fig. 3. The sensitivity graph produced by the illustrative neural network given in [1]. This shows how sensitivity to the uniform environment varies during the time course of the simulation. A sensitivity close to zero indicates that the environment is known and that the output of the network is small. Conversely, a sensitivity close to one indicates that the environment is novel and that the output is close to the sum of the intensities in the centre patch.

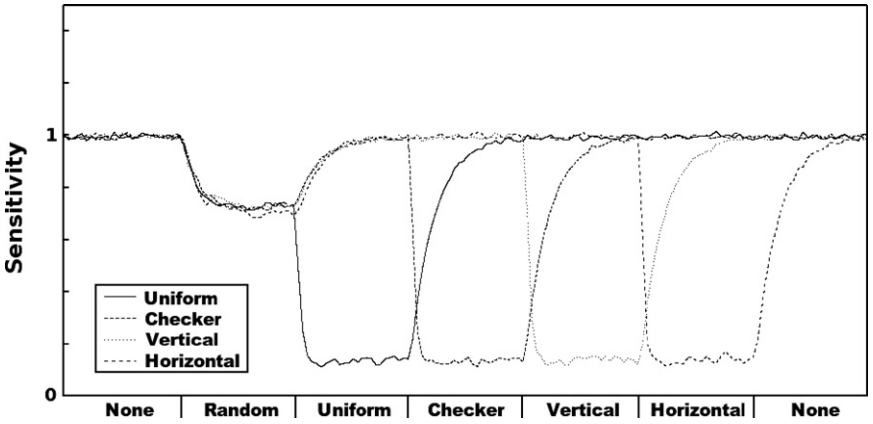

Fig. 4. The sensitivity graph produced by the illustrative neural network given in [1]. This shows how sensitivity to the uniform, checkerboard, vertical bar and horizontal bar environments vary as the network adapts to each environment during the time course of the simulation.

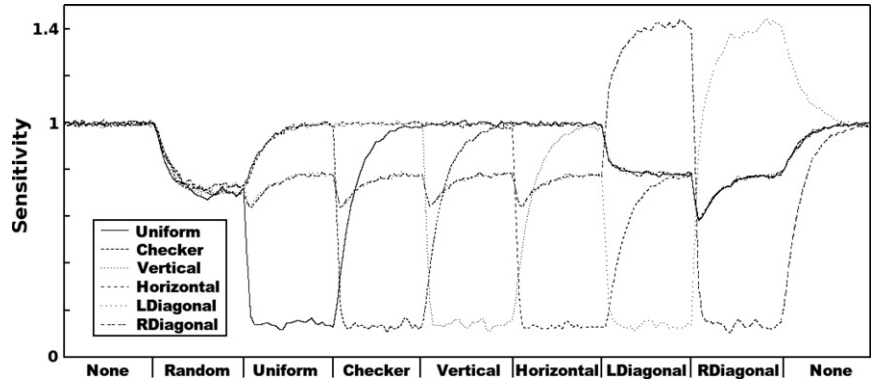

Fig. 5. The sensitivity graph produced by the illustrative neural network. In addition to the environments seen in Fig. 4, this graph shows the networks response to the left diagonal and right diagonal environments.

To explain this result, we considered the original network when adapted to a non-diagonal environment. In this case, sensitivity to diagonal environments is reduced because similarities between the diagonal and non-diagonal environments are used to form a partial prediction (this can also be seen vice versa as the network adapts to the diagonal environments). If the original network is adapted to a diagonal environment, sensitivity to the alternative diagonal environment rises above 1. This was caused by a limitation of the network in handling symmetry between environments.

3. Optimising structure using evolution

To optimise the neural network, we searched for a structure which gave the best performance according to two basic criteria of novelty detection. We wished to find a solution which (a) maximises the difference in sensitivity between known and novel environments and (b) remains at a similar level of sensitivity for all novel environments. We first attempted to construct a new structure by hand but this proved to be time consuming and none of the networks developed gave any significant improvement in performance. We then considered a genetic algorithm (GA) based approach, since such an approach is good in cases when the search space is not well understood [8]. We compared three GA's for this task; a simple textbook-based genetic algorithm, and two neuroevolution methods which are specifically designed to evolve neural network structure.

The simple genetic algorithm is based on that described by Mitchell [9]. Each gene is represented by the quadruple (*inID*, *outID*, *weight*, *type*), which is in turn encoded as a 25 character bitstring. A genome holds a collection of these genes and thus has a connection-centric view of the neural network. Point mutation and single-point crossover are both used, as well as a 'gene-replicate' and 'gene-remove' mutation (to allow the addition or removal of new structural elements).

The first neuroevolution method used was NeuroEvolution of Augmenting Topologies (NEAT), proposed by Stanley [10]. NEAT is specifically designed to evolve neural networks. New structure is introduced gradually so as minimise the dimensionality of the search space. Speciation is used to encourage diversity and help prevent bloat from occurring (solutions containing unnecessary elements). In crossover, NEAT uses historical markings to discover which genes in two genomes match and which do not, solving the competing conventions problem (whereby two identical solutions have different genetic representations). The competing conventions problem is important to solve because crossover of similar solutions with different representations is likely to produce damaged offspring [10].

We also used a variation of NEAT proposed by Whiteson et al. [12], Feature Selective NEAT (FS-NEAT). Unlike NEAT, which usually starts with a population of fully connected networks, FS-NEAT starts with a population where each network has only a single connection between a randomly chosen input and output. This allows FS-NEAT to begin its search in a space of an even lower dimensionality. FS-NEAT then proceeds in the same manner as NEAT. Experiments conducted in an autonomous car racing simulation showed that FS-NEAT was capable of outperforming NEAT in evolving solutions that both scored higher and were also less complex in terms of their structure [12].

All three approaches use the same measure of fitness, based on our performance criteria stated earlier. We define the following fitness function over N environments (not including 'none' or 'random')

$$F(c) = \sum_{i=1}^{N} \frac{1}{N-1} \left(\sum_{j=1, j\neq i}^{N} S_c(i,j) \right) - S_c(i,i) \tag{4}$$

The sensitivity of a network to a given environment E is defined as the square root of the variance of the output neuron y averaged over stimuli from environment E [1]

$$S_E = \sqrt{\langle \mathrm{var}(y) \rangle_E} \tag{5}$$

We then define $S_c(i,j)$ as the sensitivity of candidate network c to environment j when adapted to environment i. Sensitivity is scaled such that a sensitivity of 1 is defined as the sensitivity of the original neural network (described in Section 2) to each environment when in an unadapted state. To encourage sensitivity to novel environments to remain at a similar level, candidates which allow sensitivity to any environment to rise above 1 should be punished. However, such candidates should not simply be awarded a fitness of zero since they may yet evolve into good solutions. Also, sensitivity values greater than 1 that have a small distance from 1 should be awarded a higher fitness than those with a large distance. From experimentation, we found that the following non-linear adjustment gave the best results

$$S_c(i,j) = \begin{cases} 2.0 - S_c(i,j)^4 & S_c(i,j) > 1.0 \\ S_c(i,j) & \text{otherwise} \end{cases} \tag{6}$$

After this adjustment, $S_c(i,j)$ may be negative. However, the lowest fitness the network can achieve when adapted to a single environment i is constrained to 0. Thus, a network performing badly when adapted to one environment but not when adapted to another is punished for its poor performance only.

4. Results

Table 1 shows how the sensitivity of the best network evolved by the simple GA varies when it is unadapted (shown the "none" environment) and when it is adapted to the environments with perfect correlational relationships. Tables 2 and 3 show these results for the best networks evolved by NEAT and FS-NEAT, respectively. The highest scoring network overall was that found by NEAT. Fig. 6 shows the sensitivity graph produced by this best overall network.

In each experiment, 10 runs of the GA were executed, with each run performing 500 generations of evolution and returning the best network evolved during that time. The best network from the 10 runs was taken to be the best network for that method. Table 4 shows the average, best and worst performances of each GA method and Table 5 shows the average hidden node and synapse count of solutions produced by each method.

Comparing the GA approaches, we can see that the fitness of their best networks are all at a similar level. The average fitness after 500 generations was highest for NEAT

Table 1
Sensitivity of the best network evolved by the simple GA to the uniform, checkerboard, vertical bar, horizontal bar and diagonal environments as the network adapts to each environment

Environment adapted	Sensitivity to					
	Uniform	Checker	Vertical	Horizontal	LDiag	RDiag
None	0.831	0.829	0.845	0.830	0.835	0.850
Uniform	**0.134**	0.785	0.884	0.783	0.697	0.976
Checker	0.792	**0.129**	0.796	0.895	0.996	0.704
Vertical	0.881	0.777	**0.134**	0.790	0.700	0.974
Horizontal	0.793	0.899	0.796	**0.118**	0.983	0.700
LDiag	0.697	0.987	0.698	0.970	**0.138**	0.415
RDiag	0.987	0.692	0.996	0.691	0.390	**0.102**

The sensitivity to the adapted environment is highlighted in bold.

Table 2
Sensitivity of the best network evolved by NEAT to the uniform, checkerboard, vertical bar, horizontal bar and diagonal environments as the network adapts to each environment

Environment adapted	Sensitivity to					
	Uniform	Checker	Vertical	Horizontal	LDiag	RDiag
None	0.871	0.854	0.873	0.853	0.857	0.854
Uniform	**0.127**	0.902	0.894	0.896	0.988	0.605
Checker	0.893	**0.113**	0.897	0.905	0.997	0.592
Vertical	0.895	0.875	**0.112**	0.891	0.612	0.987
Horizontal	0.900	0.892	0.895	**0.117**	0.598	1.004
LDiag	0.987	0.999	0.609	0.599	**0.125**	0.506
RDiag	0.610	0.617	0.996	1.006	0.510	**0.120**

The sensitivity to the adapted environment is highlighted in bold.

Table 3
Sensitivity of the best network evolved by FS-NEAT to the uniform, checkerboard, vertical bar, horizontal bar and diagonal environments as the network adapts to each environment

Environment adapted	Sensitivity to					
	Uniform	Checker	Vertical	Horizontal	LDiag	RDiag
None	0.684	0.687	0.682	0.686	0.683	0.675
Uniform	**0.041**	1.026	0.661	1.023	0.673	1.005
Checker	1.031	**0.063**	1.046	0.675	0.319	0.690
Vertical	0.669	1.002	**0.045**	0.993	0.683	0.351
Horizontal	1.009	0.678	1.013	**0.042**	0.993	0.658
LDiag	0.672	0.340	0.682	0.990	**0.031**	0.350
RDiag	1.013	0.669	0.341	0.663	0.342	**0.044**

The sensitivity to the adapted environment is highlighted in bold.

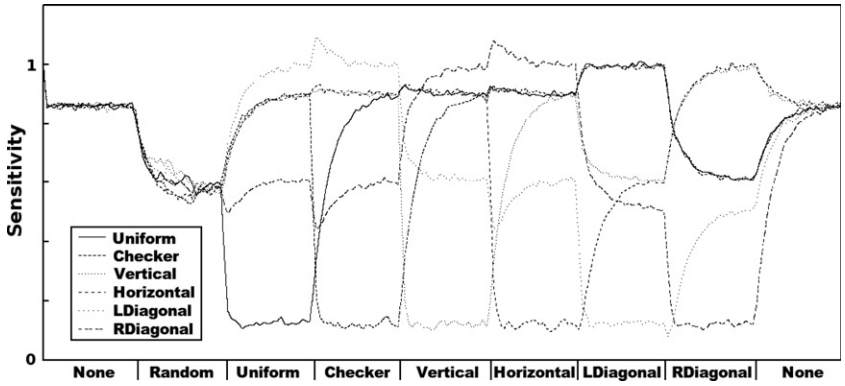

Fig. 6. The sensitivity graph produced by the best network found by NEAT.

Table 4
Average, best and worst fitness observed for each GA method over 10,500-generation runs

Method	Average fitness	Standard deviation	Best fitness	Worst fitness
Simple GA	3.537	0.220	4.122	3.405
NEAT	4.169	0.032	4.208	4.096
FS-NEAT	3.708	0.201	4.167	3.532

Table 5
Average hidden node, synapse and ineffective/unstimulated node counts for solutions produced by each GA method over 10,500-generation runs

Method	Average hidden nodes	Average modifiable synapses	Average fixed-weight synapses	Unstimulated nodes	Ineffective hidden nodes
Simple GA	7.7 (9.76)	16.5 (1.9)	4.6 (1.07)	3.1 (4.51)	6.8 (9.17)
NEAT	3.6 (2.84)	18 (2.87)	8.2 (4.02)	0.0 (0.0)	0.0 (0.0)
FS-NEAT	8.1 (4.01)	9.7 (4.69)	14.1 (5.34)	0.0 (0.0)	0.0 (0.0)

Numbers in brackets represent standard deviation. Unstimulated nodes are those which do not receive any input. Ineffective hidden nodes are those which do not influence the output neuron.

(4.169) with a standard deviation of 0.032, demonstrating a more consistent performance. Looking at the complexity of networks evolved, NEAT tended to evolve networks with fewer hidden nodes, but with a similar average synapse count to that seen for the simple GA. FS-NEAT tended to evolve networks with more fixed-weight synapses. The networks evolved by the simple GA also showed evidence of bloat in that they had both nodes with no inputs (unstimulated nodes) and hidden nodes which did not connect to, or influence in any way, the output neuron. However, such obsolete structure can easily be pruned from these networks.

In most practical applications, data is likely to contain a noise component. Therefore, a novelty detector used in such applications should be resilient to noise. To investigate this, we tested the performance of the best evolved network when independent noise was introduced to each input. The noise component was a value drawn independently from a standard normal distribution and scaled by a constant factor k. We varied k from 0.0 to 5.0, with a step size of 0.2. Fig. 7 shows how the fitness of both the original illustrative neural network and the best evolved network varies as noise is introduced. This reduces in a non-linear fashion for both networks as the noise factor k is increased. The best evolved network is able to distinguish between novel and known environments until $k = 0.8$, after which it struggles to reliably differentiate between the diagonal environments. At $k = 1.6$, the best evolved network is unable to distinguish between any of the environments. The original network is unable to distinguish between environments at $k = 2.2$.

5. Discussion

The networks evolved by the GA methods show a more consistent handling of a broader range of environments than either the example neural network given by Hosoya et al. [1] or any of the networks we designed by hand. The network with the highest fitness, found using NEAT, has a complexity similar to that of the example network. For most novel environments, this networks sensitivity remains at a similar level throughout the simulation. Unlike the example network, this network does not see a dramatic increase in sensitivity to one diagonal environment when adapted to the alternative diagonal environment. This is important as it ensures a more consistent

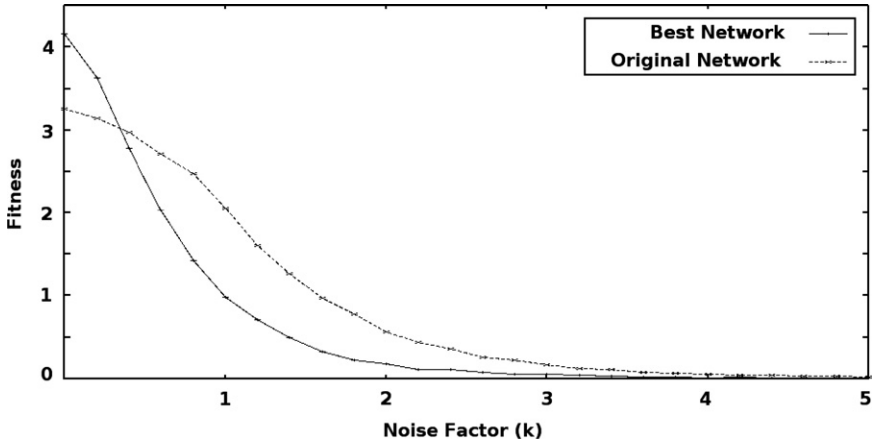

Fig. 7. Fitness of the best evolved network as noise is applied to its inputs, with noise factor k varied from 0.0 to 5.0.

separation between novel and known environments. However, a drop in sensitivity to the diagonal environments is still observed when the network is adapted to non-diagonal environments, and vice versa. This is again caused by the network being able to form partial predictions of stimuli due to similarities between environments.

Interestingly, the structure of the best network has demonstrated, from a predictive coding standpoint, a change in function. The fixed-weight synapses between the centre patch inputs and the output neuron have been removed and a new fixed-weight synapse introduced connecting a single peripheral input to the output neuron. Thus, the network has changed from predicting the sum of the centre patch to predicting the selected periphery input. However, from a novelty detection perspective this network retains the intended function of indicating the novelty of a given input vector. The high-scoring candidates from all three GA approaches demonstrate similar changes in network structure. This may indicate that a network which preserves the function of predicting the sum of the centre patch does not exist.

A surprising result is the comparatively high best fitness score achieved by the simple GA. For this problem, the best network evolved by the simple GA was of a similar fitness to the best networks evolved by both NEAT and FS-NEAT. This is despite the relative complexity of these two neuroevolution techniques, compared to the simple GA. Improvements can easily be made to the simple GA, such as adding new nodes using a similar method to that used by NEAT to reduce the observed bloating of networks. Whilst NEAT has been demonstrated to give a more reliable performance, investigating the effect of such improvements to the performance of the simple GA would be instructive.

The best network presented in Section 4 has been shown to cope reasonably well with noise. At $k = 0.8$, the variance of the noise component was 0.64 times the variance of the signal component. Despite this, the network was still capable of distinguishing between known (adapted) and novel (unadapted) environments. However, with increasing noise, the network is shown to perform worse than the original network in terms of the fitness criteria defined in Section 3. This demonstrates that performance over noisy data should be considered by the GA approaches when searching for new neural network structures.

6. Conclusions and future work

We have described a neural network used by Hosoya et al. [1] to illustrate dynamic predictive coding and identified a limitation of this network. For the task of optimising the structure of the network, we have demonstrated that whilst NEAT outperforms two other evolutionary algorithms, it does not produce a solution which is significantly better than that produced by a simple genetic algorithm. The optimised network evolved by NEAT distinguishes more consistently between a broader range of environments than either the original neural network or any of the networks we designed by hand. It also performs well when a low level of noise is introduced but its performance degrades quickly as this noise increases. This is because noise sensitivity was not part of the fitness criteria used by the GA approaches. Since we plan to test this approach to novelty detection on data that is inherently noisy, this will need to be addressed in future work.

In order to extend this work, we plan to investigate using a neural network, capable of dynamic predictive coding, for novelty detection over recorded weather data. A vector of metrics would be passed to the network, which would then learn the correlational relationships between those metrics and detect novelty when these relationships change. We consider it likely that the structure of the neural network used will need to be evolved to give optimal performance in this task, and this process of evolution will need to take into consideration noise applied to the data.

References

[1] T. Hosoya, S.A. Baccus, M. Meister, Dynamic predictive coding by the retina, Nature 436 (2005) 71–77.
[2] J. Jiang, Image compression with neural networks – a survey, Signal Process, Image Commun. 14 (1999) 737–760.
[3] M. Markou, S. Singh, Novelty detection: a review part 1: statistical approaches, Signal Process. 83 (12) (2003) 2481–2497.
[4] M. Markou, S. Singh, Novelty detection: a review part 2: neural network based approaches, Signal Process. 83 (12) (2003) 2499–2521.
[5] S. Marsland, U. Nehmzow, J. Shapiro, Detecting novel features of an environment using habiutation, in: Proceedings of the Simulation of Adaptive Behaviour, MIT Press, Cambridge, MA, 2000.
[6] S. Marsland, U. Nehmzow, J. Shapiro, On-line novelty detection for autonomous mobile robots, Robo. Auto. Syst. 51 (2–3) (2005) 191–206.
[7] S. Marsland, J. Shapiro, U. Nehmzow, A self-organising network that grows when required, Neural Netw. 15 (8-9) (2002) 1041–1058.
[8] M. Mitchell, An Introduction to Genetic Algorithms, 6th ed., MIT Press, Cambridge, MA, 1999.
[9] T. Mitchell, Machine Learning, int. ed., McGraw-Hill Higher Education, 1997.
[10] K.O. Stanley, Efficient evolution of neural networks through complexification, Ph.D. thesis, University of Texas at Austin, August 2004.
[11] P. Stirling, The Synaptic Organization of the Brain, chap. Retina, 3rd ed., Oxford University Press, Oxford, 1990, 170–213.
[12] S. Whiteson, P. Stone, K.O. Stanley, R. Miikkulainen, N. Kohl, Automatic feature selection in neuroevolution, in: Proceedings of the Genetic and Evolutionary Computation, ACM Press, 2005.