



Performance-guided neural network for rapidly self-organising active network management

Sin Wee Lee^{a,*}, Dominic Palmer-Brown^a,
Christopher M. Roadknight^b

^a*Leeds Metropolitan University, Computational Intelligence Research Group, Beckett Park,
Leeds LS6 3QS, UK*

^b*BTexact Technologies, BT Adastral Park, Martlesham Heath, Ipswich IP5 3RE, UK*

Abstract

We present a neural network for real-time learning and mapping of patterns using an external performance indicator. In a non-stationary environment where new patterns are introduced over time, the learning process utilises a novel *snap-drift* algorithm that performs fast, convergent, minimalist learning (*snap*) when the overall network performance is poor and slower, more cautious learning (*drift*) when the performance is good. Snap is based on a modified form of Adaptive Resonance Theory (CGIP 37(1987)54); and drift is based on Learning Vector Quantization (LVQ) (Proc. IJCNN 1(1990a)545). The two are combined within a semi-supervised learning system that shifts its learning style whenever it receives a significant change in performance feedback. The learning is capable of rapid re-learning and re-stabilisation, according to changes in external feedback or input patterns. We have incorporated this algorithm into the design of a modular neural network system, Performance-guided Adaptive Resonance Theory (P-ART) (Proc. IJCNN 2(2003)1412; *Soft computing systems: Design, Management and application*, IOS Press, Netherland, 2002; pp. 21–31). Simulation results show that the system discovers alternative solutions in response to significant changes in the input patterns and/or in the environment, which may require similar patterns to be treated differently over time. The simulations involve attempting to optimise the selection of network services in a non-stationary, real-time active computer network environment, in which the factors influencing the required selections are subject to change.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Snap-drift; Adaptive resonance theory; Learning vector quantization; P-ART Performance indicator

* Corresponding author. Tel.: +44-113-283-2600.

E-mail addresses: s.w.lee@leedsmet.ac.uk (S.W. Lee), d.palmer-brown@leedsmet.ac.uk (D. Palmer-Brown).

URL: <http://www.lmu.ac.uk/ies/comp/research/cig/>

1. Introduction

1.1. Adaptive resonance theory (ART) network

The ART network is an artificial neural network (ANN) that was first introduced by Stephen Grossberg to overcome the *stability–plasticity* dilemma [16,17]. Recent developments have led to various types of ART networks. ART1 [5] self-organises an arbitrary number of binary input sequences into a set of output recognition categories. A second type, ART2, does the same for either binary or analogue inputs [6]. Subsequent instantiations and development of the theory, such as ART3 [7] are used to implement parallel searches of compressed or distributed recognition codes (output categories) in a neural network hierarchy.

Following the successful implementation of the theory in real-time applications, further development of the theory has seen the creation of ART2-A [8], which is 2 or 3 orders of magnitude faster than ART2. Fuzzy ART [9], the fuzzy extension of ART, incorporates computations from fuzzy set theory into the ART1 neural network. Extensions to ART networks to allow supervised learning have also gained much interest [2,23,29,30]. For example, ARTMAP [10] and Fuzzy ARTMAP [11] have proved successful at autonomously learning to classify patterns based on predictive success.

1.2. The ART1 architecture

ART1 networks are capable of fast and stable learning by categorising arbitrary binary input patterns using the basic principles of self-organization. The ART1 network (see Fig. 1) consists of 3 layers: the input layer (F0), the comparison layer (F1) and

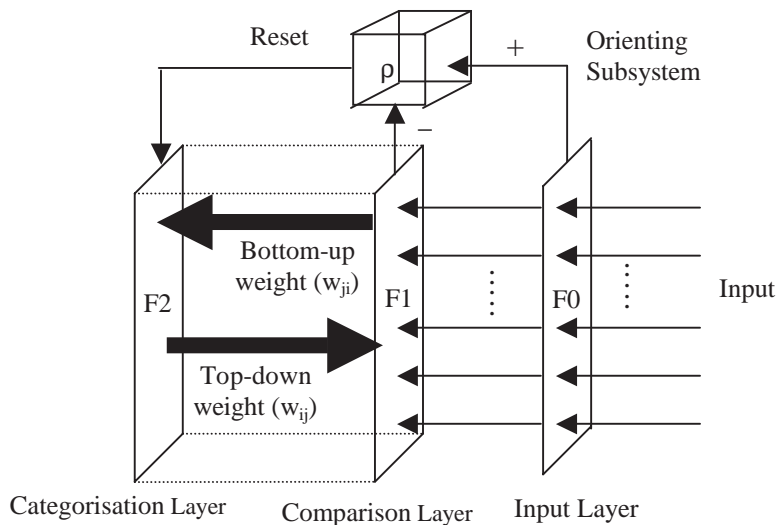


Fig. 1. Architecture of an ART1 network.

the categorisation layer (F2) with N , N , and M number of nodes, respectively. Each node of the input layer is connected via non-modifiable links to its corresponding node in the comparison layer (there is a one to one mapping between F0 and F1 nodes). The F1 and F2 layers are interconnected using bottom-up and top-down modifiable weighted links that are adapted during the learning stage.

1.3. The ART1 learning principles

The learning process of the network can be described as follows: Upon the presentation of a binary input pattern I ($I_j \in \{1, 0\}, j = 1, 2, 3, \dots, N$), the network attempts to categorize it by comparing it against the stored knowledge of the existing categories of each F2 node. This is achieved by calculating the bottom-up activation, which can be expressed as

$$T_i = \frac{|w_i \cap I|}{\beta + |w_i|}, \quad (1)$$

where β is the constant that enables the larger magnitude prototype (weights encoding a minimal set of features that describe all input or ‘member’ patterns allocated to that category node) vector to be selected when there exist multiple prototype vectors that are subset of the binary input pattern.

The F2 node with the highest bottom-up activation, i.e. $T_I = \max\{T_i | I = 1, 2, \dots, M\}$ is then selected. If a category is found with the required matching level, known as the *vigilance level* and represented by the vigilance parameter ρ where $0 < \rho < 1$ and expressed by (2), then F2 node J will enter into a resonant state whereby it learns by modifying its prototype (weights encoding a minimal set of features that describe all input or ‘member’ patterns allocated to that category node), to retain only the critical features for the selected output category. This adjustment process is expressed by (3)

$$\frac{|w_I \cap I|}{|I|} \geq \rho, \quad (2)$$

$$w_{iJ}^{(\text{new})} = \eta(w_{iJ}^{(\text{old})} \cap I) + (1 - \eta)w_{iJ}^{(\text{old})}, \quad (3)$$

where η is the learning rate ($0 < \eta < 1$). All other weights in the network remain unchanged.

If no existing matching prototype is found, i.e. when the stored w_J do not match the input sufficiently, then the winning F2 node is reset and another F2 node with the highest activations is selected based on the similarity between its prototype and the current input, and so on. When no corresponding output category (F2 node) can be found, the network considers the input pattern as novel, and generates a new output category that learns the current input. Essentially, it is a fast adaptive form of competition-based learning [12].

2. The British telecom (BT) application

2.1. Application layer active network (ALAN)

An ideal computer network infrastructure would be one that is as simple as possible, removing duplication of management overhead, minimising signalling and moving towards an optimum (hands-off) network. ‘Active Networking’ [32] aims to achieve this. It gives users the ability to load software components onto network devices dynamically without explicit reference to any third party. There are three types of active networks:

- (1) *Capsule*—This approach is used to enable the active service codes to run on the network devices, such as servers, that the packets encounter.
- (2) *Programmable*—This approach allows the clients to download their own active service codes onto network devices before using their application [4].
- (3) *Active bridging*—This approach allows the network operators¹ the freedom to choose appropriate active service codes of their own [1]. Despite the difficulties of the approach [27], it has highlighted the important requirements for a feasible active network and encouraged other researchers to develop better alternatives to resolve them.

The ALAN architecture was first proposed by Fry and Ghosh [15] to enable the user to supply JAVA-based active-service codes known as *proxylets* that run on an edge system (Execution Environment for Proxylets—EEPs) provided by the network operator. The purpose of the architecture is to enhance the communication between servers and clients using the EEPs that are located at optimal points of the end-to-end path between the server and the clients, without having to deal with the current system architecture and equipment. This approach relies on redirecting selected request packets into the EEP, where the appropriate proxylets can be executed to modify the packet’s contents without impacting on the router’s performance.

2.2. Automated active network management using distributed genetic algorithms

In the original ALAN proposal, the management system supports conventional management agent interfaces [25,26] that respond to high-level instructions from the system operators. However, since ALAN provides active services that are unbounded in both scale and functions, with an enormous range of services being developed and evolved at an unprecedented rate, it is necessary to combine the active services with a highly automated and adaptive management and control solution. Recently, a novel adaptive distributed Genetic Algorithm approach (GA) to this problem of automated network management was introduced by Marshall and Roadknight, from British Telecom (BT) Research Laboratories, and it solves the problem with some success.

¹ Network operators are those companies (such as BT, NTL) who control the public network and allow Internet Service Providers (ISPs) (such as Freeserve, BT, AOL) to provide additional Internet services.

The algorithm was applied to the adaptive management solution and to the differentiated quality of service mechanism defined by Marshall & Roadknight [28] for ALAN and has shown promising results with respect to increasing the performance of the ALAN network. In the work presented here, P-ART is used as a means of finding and optimising a set of conditions that produce optimum proxylet selection in the Execution Environment for Proxylets (EEP), which contains all the frequently requested proxylets (services). P-ART learns the conditions from user request patterns, and periodic performance feedback influences the learning. The learned conditions are used by P-ART to select proxylets.

3. The performance-guided ART (P-ART) architecture

3.1. The P-ART system

The ART1 network tends to organise itself into a stable state due to fast learning, resulting in weights that can no longer adapt. In the original ART algorithm there is no provision for external feedback to improve network performance once it has stabilised. Although vigilance can be varied, the highly convergent learning style inherently supports continual reduction of a winning node's prototype so that it contains only the minimal set of elements that are common to all input patterns for which it is the winning node.

P-ART introduces performance feedback, as described later in this paper, which encourages winning nodes to increase information encoded within the weights as well as reduce or eliminate it, by altering the learning style. Before discussing this further, the overall P-ART architecture is described.

The P-ART network proposed is a modular, multi-layered architecture as shown in Fig. 2. On the presentation of an input pattern at the input layer F_{01} , the dP-ART will learn to group the input patterns according to their general features using the novel learning principles developed in this work, known as the 'snap-drift' algorithm. However, the standard matching and reset mechanism of ART [13] is used. The size of the input layer, F_{01} , is fixed and equal to the request pattern width of 50 bits, which is determined by the proxylet information available, as described later. There could in theory be a significant increase in proxylet types without any increase in the input size.

The classic ART reset is deployed, whereby if no existing matching prototype is found, i.e. when the stored pattern prototypes are not a good match for the input, the winning F_{21} node is reset and another F_{21} node is selected. When no corresponding output category can be found, the network considers the input as novel, and generates a new output category that learns the current input pattern. The three winning F_{21} nodes ($D = 3$) whose prototypes best match the current input pattern, are the input data to the sP-ART module for selecting one appropriate output type (called a proxylet in the target application).

For the purpose of selecting the required proxylet, the proxylet type information indicated by the P-ART references pre-trained locations on the Kohonen Self-organising Map (SOM) [19,20], which represents specific proxylets. If the referenced proxylet

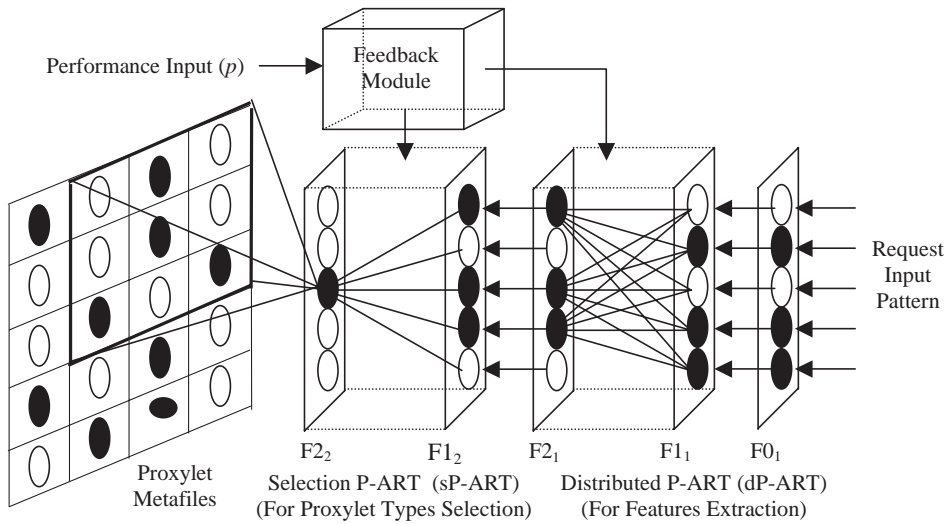


Fig. 2. Architecture of the P-ART network.

is unavailable, one of its nearest neighbours is selected (the most similar alternative available). Hence, the number of F_{2_2} nodes is the number of proxylet types (maximum of 100 in this case), which will always be less than the number of proxylets on the active network, since some are of the same type, and is independent of the number of nodes in the active computer network.

A non-specific performance measure is used because, as in many applications, there are no specific performance measures (or external feedback) in response to each *individual* output decision. The measure is used to encourage or discourage reselection of outputs (proxylet types) to occur in order to improve the performance of the neural system. A summary of the steps that occur in P-ART is shown in Fig. 3.

3.2. The dP-ART learning principles

The purpose of the dP-ART module is to learn key features, which can then be used by sP-ART to learn to select an appropriate output that references a desired location on the application-specific SOM network. In dP-ART, on presentation of a binary input pattern, I , the network attempts to categorise it according to the stored knowledge (weights) of the existing distributed output categories of the F_{2_1} layer. This is achieved by calculating the bottom-up activation, using (1).

As this architecture is based on a distributed ART (dART) [14], there is more than one winning node, in this case $D=3$. The three F_{2_1} nodes with the highest bottom-up activations are selected.

If a distributed output category is found with the required matching level, using (2), as in ART, where vigilance parameter $0 < \rho < 1$, the three F_{2_1} nodes with the highest activation would enter into a resonant state and learn by modifying their weights to

Initialise parameters.

For each new input pattern

For the top D distributed winning nodes:

1. Find the candidate F2₁ node to learn the current input pattern i.e. F2₁ (which is not inhibited) whose prototype is most similar to the current input pattern.
2. Test reset condition:
 - a. If reset is true, then the current candidate F2₁ node is rejected; inhibit node and return to 1.
 - b. If reset is false, then current candidate F2₁ node is accepted for learning; proceed.
3. Learning:
 - a. Weights of dP-ART are adapted according to the snap-drift learning style indicated by the external performance feedback value.

Process the output pattern of F2₁ as an input pattern to F1₂

Find winning node in F2₂, by the same procedure as above.

Adjust sP-ART weights using the snap-drift determined by the external performance feedback level.

Fig. 3. Summary of the main steps in P-ART.

keep only the common features for the selected output category. However, to allow sensitivity to performance, we introduce a new form of learning that is dependent upon a performance feedback parameter, as shown in (4)

$$w_{ij}^{(\text{new})} = (1 - p)(I \cap w_{ij}^{(\text{old})}) + p(w_{ij}^{(\text{old})} + \beta(I - w_{ij}^{(\text{old})})), \quad (4)$$

where, $w_{ij}^{(\text{old})}$ is the top-down weights vectors at the start of the input presentation, p the performance feedback parameter, I the binary input vectors, β the ‘drift’ constant.

In general, (4) can be stated as

$$w_{ij}^{(\text{new})} = \alpha(\text{fast_learning ART}) + (1 - \alpha) (\text{LVQ}), \quad (5)$$

where α is determined by performance feedback. So, in one sense P-ART is like ART, an unsupervised learning method. However, unlike ART, its dynamic behaviour is partially governed by performance feedback from the environment. To achieve this, P-ART combines minimalist ART learning with Learning Vector Quantization (LVQ) [18]. By substituting p in (4) with 0 for poor performance, (4) simplifies to

$$w_{ij}^{(\text{new})} = (I \cap w_{ij}^{(\text{old})}). \quad (6)$$

This invokes fast minimalist learning, causing the top-down weights to reach their new asymptote on each input presentation:

$$w_j \rightarrow I \cap w_j^{(\text{old})}. \quad (7)$$

In contrast, for excellent performance, where $p = 1$, (4) simplifies to

$$w_{ij}^{(\text{new})} = (w_{ij}^{(\text{old})} + \beta(I - w_{ij}^{(\text{old})})). \quad (8)$$

This causes a simple form of clustering or LVQ at a speed determined by β . In practice, the learning is usually a composite of the two forms of learning. It is assumed that there is a considerable interval between updates of p during which time new (previously unseen) input patterns are likely to appear. Eq. (8), or indeed (4) when performance is not perfect, enables the top-down weights to drift towards the input patterns. With alternate episodes of $p = 0$ and 1, the characteristics of the system's learning will be the joint effects of the (6) and (8). This enables the system to learn using fast and convergent snap learning when the performance is poor, yet drift towards and therefore take into account new input patterns when the performance is good.

If no matching prototype exists, i.e. when the stored w_j does not match the current input pattern sufficiently, then the winning $F2_1$ node is reset and another $F2_1$ node with the highest T_i is selected, as in ART. When no corresponding distributed output category can be found, the network considers the input as novel, and expands by generating a new distributed output category with an associated set of weights. This new output node (category) is then associated with the current input vector by making its weight vector equal to the input vector.

The bottom-up learning in P-ART is a normalised version of the top-down learning:

$$w_{jl}^{(\text{new})} = (1 - p) \frac{I \cap w_{jl}^{(\text{old})}}{|I \cap w_{jl}^{(\text{old})}|} + p \left(w_{jl}^{(\text{old})} + \beta \frac{I \cap w_{jl}^{(\text{old})}}{|I \cap w_{jl}^{(\text{old})}|} \right), \quad (9)$$

where $w_{jl}^{(\text{old})}$ the bottom-up weights of the network at the start of the input presentation.

At the beginning of the input presentations, the bottom-up weights w_{ji} are assigned initial values corresponding to the initial top-down weights w_{ij} values using (10)

$$w_{ji}(0) = \frac{w_{ij}(0)}{1 + N}, \quad (10)$$

where N is the Number of input nodes.

By selecting this small initial value of w_{ji} , the network is more likely to select a previously learned category node that to some extent matches the input vector, rather than an uncommitted node.

During the learning phase, if the network encountered poor performance, by substituting $p = 0$ in (9), the bottom-up learning of the network can be illustrated as follows:

$$w_{jl}^{(\text{new})} = \frac{I \cap w_{jl}^{(\text{old})}}{|I \cap w_{jl}^{(\text{old})}|}. \quad (11)$$

This forces fast, convergent *snap* learning.

In contrast, if the network encountered perfect performance after a considerable interval, (9) would simplify to

$$w_{jl}^{(\text{new})} = p \left(w_{jl}^{(\text{old})} + \beta \frac{I - w_{jl}^{(\text{old})}}{|I - w_{jl}^{(\text{old})}|} \right). \quad (12)$$

This will result in the weights *drifting* towards input patterns encountered at subsequent time steps.

Drift alone will only result in slow (depending on β) reselection over time, thus forming modified or new representations of new input patterns without a radical set of reselections for previously learnt patterns. Conversely, snapping entails rapid reselection for a proportion of patterns as a fast response to significant changes, in the input vectors (requests) and/or the environment. In this way, the same requests can be treated differently by snapping from a new position in weight space that has been set up by drift, which effectively moves the system out of a local minimum. Thus, a new category node selection may occur for one of two reasons: (a) as a direct result of the drift itself; or (b) as a result of the drift enabling a further snap to occur (since drift has moved weight templates away from convergence) when performance p decreases.

3.3. The sP-ART learning

The distributed output categories produced by the dP-ART act as inputs to the sP-ART. The architecture of the sP-ART is the same as that described in Section 3.1 with one exception; only the $F2_2$ node with the highest activation learns. Each output node of the sP-ART represents a set of available application-specific groupings (in this case proxylet types). These proxylet types have been used to generate training data for the SOM so that the SOM can be independently trained to form regions whereby similar proxylets are organised in adjacent nodes on the map. This allows each output node of the sP-ART to be ‘hardwired’ pointers to regions of the SOM. The task of the sP-ART is therefore to learn to associate the correct group of input patterns with an output node that is hardwired to a region of the SOM that represents the best proxylet selections for the group. The effect of learning and relearning within the sP-ART module is that specific output nodes will relate different groups of input patterns to specific regions of the SOM until the performance feedback indicates that it is indexing the SOM region corresponding to the most appropriate proxylets.

3.4. The performance feedback

The external performance feedback into the P-ART system reflects the performance requirement in different circumstances. Various performance feedback profiles within the range $\{0, 1\}$ are fed into the network to evaluate the dynamics, stability and performance responsiveness of the learning. Initially, some very basic tests with performances of 1 or 0 were evaluated in a simplified system [21,22]. The simulations presented below involve computing the performance based on a parameter associated with the winning output neuron. Ultimately, a realistic commercial external performance feedback criterion, obtained from British Telecom (BT), will be established, to evaluate the improvement in performance of the network learning under realistic external performance feedback criteria. In the BT application, functions which contribute to good/poor performance include latencies for user service request with differing time to live, dropping rate for request with differing time to live, different charging level according to quality of service, and so on.

4. Simulations

This section presents a number of simulations performed and an evaluation of the results produced using P-ART to assess the behavioural characteristics of the *snap-drift* algorithm. The current BT active computer network implementation is a test bed platform with up to 16 servers. Proxylets have been written to do a variety of tasks including active compression, multimedia streaming, dynamic resource caching/discovery, multicast and video conferencing [24].

4.1. Input encoding

This section describes the encoding method used to represent user request data within the P-ART input patterns. Coarse coding is used to represent proportional differences between numeric data encoded within the input patterns, e.g. the representation of the value 15 must be closer in input space to the representation of value 20 than that of say 30. Likewise, the representation of the value 100 must be more similar to that of 120 than that of 150, and so on. Table 1 shows the realistic range for each of the request properties.

The training patterns consist of 100 input vectors. For simplicity, the input pattern is arranged in a 10×5 matrix. Whilst each row represent the properties of a user request, such as bandwidth, time, file size, loss and completion guarantee, the system treats each input pattern as a whole. These test patterns were presented in random order for 80 epochs. This on-line continuous random presentation of test patterns simulates possible real world scenarios whereby the order of patterns presented is random so that a given network request might be repeatedly encountered while others are not encountered at all.

4.2. Weights initialisation

The weights are calculated as floating point and are initialised at the beginning of the simulations. Top-down weights w_{ji} are set randomly to either 0 or 0.99.

$$w_{ij}(0) = [0, 0.99]. \quad (13)$$

Thus, a simple distributed affect will be generated at the output layer of the network, with different patterns tending to give rise to different activations across F_2 .

Table 1
Value ranges of a user request attributes

Properties	Range
Bandwidth	10 Kb/s \rightarrow 2000 Kb/s
Time	1 ms \rightarrow 1000 ms
Loss	20% \rightarrow 60%
Cost	0.1 p \rightarrow 100 p
Completion guarantee	40% \rightarrow 100%

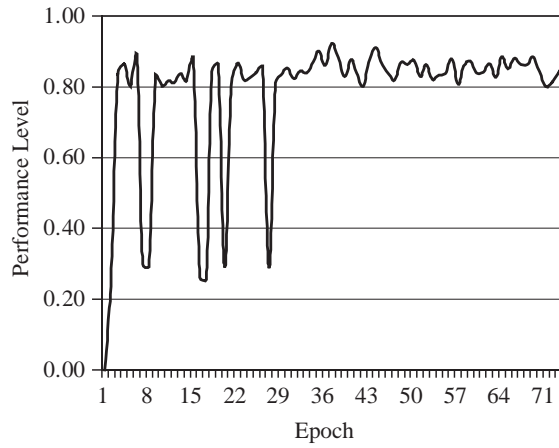


Fig. 4. Performance levels of the P-ART network.

The bottom-up weights w_{ij} are assigned initial values corresponding to the initial values of the top-down weights w_{ji} . This is accomplished by

$$w_{ji}(0) = \frac{w_{ij}(0)}{1 + N}, \quad (14)$$

where N is the number of input nodes.

4.3. Performance calculation

The performance of the network is calculated after every simulation epoch. This is done by calculating the average of the proxylet property that is taken as the selection criterion over a particular simulation epoch. Then, the performance is used as the performance feedback for the next simulation epoch. So, the calculation of the performance can be summarised as follows:

$$\text{Performance } p = \text{average}(\text{proxylet_property}(\text{epoch})). \quad (15)$$

4.4. Results

In Fig. 4, we show the performance calculated across the simulation epochs. The network starts with low performance and the performance feedback is calculated and fed into the dP-ART and sP-ART after every simulation epoch, to be applied during the following epoch. Epochs are of fixed length for convenience, but could be any length. The aim in this simulation is to encourage the system to select high bandwidth proxylet types, so bandwidth is the property.

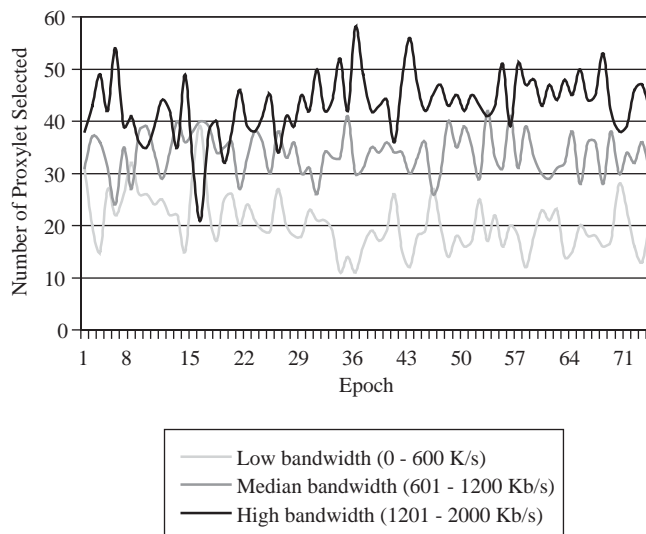


Fig. 5. Selection frequency of the proxylet types.

Fig. 5 shows the selection frequency of the proxylet type. In this case, we have the following bandwidth ranges:

- (i) Low bandwidth proxylet: $0 \rightarrow 600$ Kb/s.
- (ii) Median bandwidth proxylet type: $601 \rightarrow 1200$ Kb/s.
- (iii) High bandwidth proxylet type: > 1200 Kb/s.

At the first epoch (refer to Fig. 4), the performance is set to 0 to invoke fast learning. A further snap occurs in epoch 7 since low performance has been detected. Note that during epochs 7 and 8, there is a significantly higher selection of high bandwidth proxylet types, caused by the further snap and continuous new inputs that feed into the network. As a result, performance has been significantly increased at the start of ninth epoch.

At epochs 16, 20 and 27, in Fig. 4, there is a significant decrease in performance. As illustrated in Fig. 5, this is caused by a significant increase in the selection of low-bandwidth proxylet types and a decrease in high-bandwidth proxylets. This is due to the drift that has occurred since the last snap, with a number of new patterns still appearing for the first time. The performance-induced snap takes the weight vectors to new positions. Subsequently, a similar episode of decreased performance occurs, for similar reasons, and a further snap in a different direction of weight space follows, enabling reselections, resulting in improved performance.

At the 28th epoch, where $p = 0.8121$, the performance has stabilised around the average performance of 0.85. At this stage, most of the possible input patterns have been encountered. Until more new input patterns are generated and introduced or there

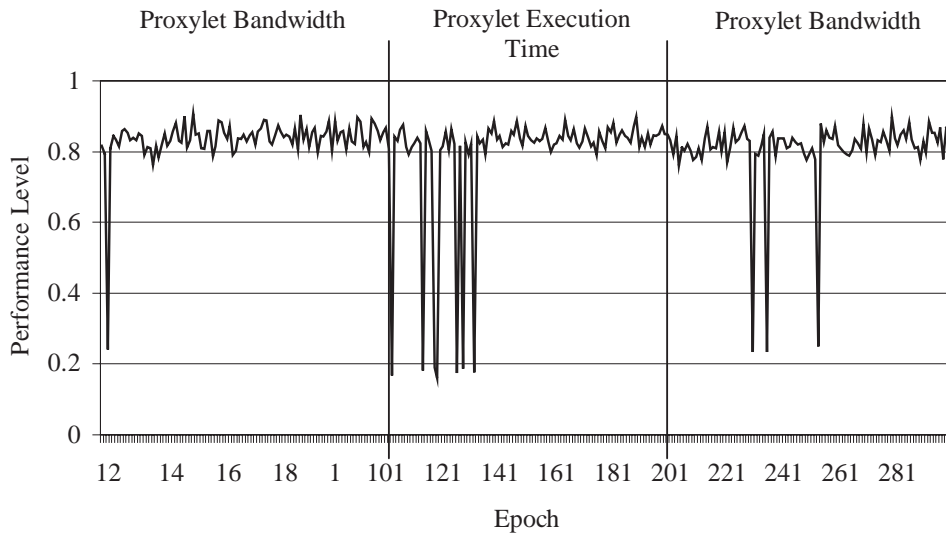


Fig. 6. Performance level of the P-ART system with alternate episodes of performance criteria.

is a change in the selection requirements that causes a dip in performance, the network will maintain this high level of performance. As shown in Fig. 6, the average proxylet execution time is introduced into the performance criterion calculation to encourage the selection of high execution time proxylet types. In this case, we have the following execution time bands:

- (i) Short execution time proxylet: $1 \rightarrow 300$ ms.
- (ii) Median execution time proxylet type: $301 \rightarrow 600$ ms.
- (iii) Long execution time proxylet type: > 600 ms.

This criterion is fed into the P-ART at every 100th epoch. When the new performance criterion is introduced, rapid reselection of a proportion of the patterns occurs in response to the significantly changed situation, followed by stabilisation. Subsequently, when the average proxylet bandwidth is reintroduced into the system, a further snap will occur.

Other parameters such as cost and file size will be added to the performance calculation to produce a more realistic simulation of network circumstances in the future. The performance of P-ART and the existing GA approach mentioned in Section 2.2, cannot be directly compared as the former is a ‘pull’ method for selecting correct proxylets and the latter is a ‘push’ method for placing proxylets in efficient locations. There is however some scope for using both methods where the GA proxylet placement algorithm optimises the proxylet groupings, thereby limiting the scope of the P-ART selection task.

5. Conclusions and future work

A multi-layer neural system containing modules that combine ART style learning with LVQ according to performance feedback has been proposed. It is capable of stable learning of the network input request patterns in real-time and is able to map them onto the appropriate proxylets available to the system. The simulations have shown the plausibility of the ‘snap-drift’ algorithm, which is able to provide continuous real-time learning in order to improve the system’s performance, based on the external performance feedback. The real-time properties have been confirmed by the results obtained from the experiments performed using P-ART with a number of performance feedback scenarios.

On-going work is exploring performance behaviour under different environmental change. Investigations and simulations will be carried out to assess P-ART’s behaviour in response to user requests and environmental situations that model known commercial circumstances, e.g. when bandwidth has more priority than the cost during video streaming, or when completion guarantee has higher priority during data transfer. This will test the ability of the learning to respond to non-linear dependencies that are changing through time.

Currently, the proxylet SOM adopts the original short offline training algorithm first proposed by Kohonen [19,20]. It may be feasible integrate the *snap-drift* algorithm into the SOM learning to provide fast, on line training. This will also provide an opportunity to evaluate the plausibility of the *snap-drift* algorithm in the context of different ANN architecture.

To assess the wider applications of the P-ART system, experiments are proposed on a different problem domain such as real-time optical character recognition (OCR) and natural language processing (NLP), to enable further understanding of the network’s dynamic behaviour and potential. OCR and NLP contain some well-understood problems that have been addressed with many types of ANNs [3,31] and such an assessment will allow a comparative analysis to be performed and thus provide insight into the computational power and limitations of the system.

Acknowledgements

This work is funded by British Telecom (BT) Laboratories. We would like to thank Ian Marshall for his supports and comments.

References

- [1] D.S. Alexander, Active bridging, *Comp. Com. Rev.* 26 (2) (1996) 101–111.
- [2] Y.R. Asfour, et al., Fusion ARTMAP: a neural network architecture for multi-channel data fusion and classification, *Proceedings of the WCNN’93*, pp. 210–215.
- [3] S. Barker, H. Powell, D. Palmer-Brown, Size invariant attention focusing (with ANNs), *Proceedings of the ISMIP’96*.
- [4] A.T. Campbell, et al., Open signalling for ATM, internet and mobile network, *Comp. Com. Rev.* 29 (1) (1999) 97–108.

- [5] G.A. Carpenter, S. Grossberg, A massively parallel architecture for a self-organising neural pattern recognition machine, *CGIP* 37 (1987) 54–115.
- [6] G.A. Carpenter, S. Grossberg, ART2: self-organization of stable category recognition codes for analogue pattern, *Appl. Opt.* 26 (1987) 4919–4930.
- [7] G.A. Carpenter, S. Grossberg, ART 3: hierarchical search using chemical transmitter in self-organizing pattern recognition architectures, *Neural Networks* 3 (4) (1990) 129–152.
- [8] G.A. Carpenter, S. Grossberg, D.B. Rosen, ART 2-A: an adaptive resonance algorithm for rapid category learning and recognition, *Neural Networks* 4 (1991) 493–504.
- [9] G.A. Carpenter, S. Grossberg, D.B. Rosen, Fuzzy ART: fast stable learning and categorization of analogue pattern by an adaptive resonance system, *Neural Networks* 4 (1991) 759–771.
- [10] G.A. Carpenter, S. Grossberg, J.H. Reynold, ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural networks, *Neural Networks* 4 (1991) 565–588.
- [11] G.A. Carpenter, et al., Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analogue multidimensional maps, *IEEE Trans. Neural Networks* 3 (5) (1992) 698–713.
- [12] G.A. Carpenter, S. Grossberg, The art of adaptive pattern recognition by a self-organising neural networks, *IEEE Comp.* 21 (3) (1988) 77–88.
- [13] G.A. Carpenter, S. Grossberg, Search mechanism for adaptive resonance theory (ART) architecture, *Proceedings of the IJCNN'89*, Vol. 1, 1989, pp. 201–205.
- [14] G.A. Carpenter, Distributed learning, recognition, and prediction by art and artmap neural networks, *Neural Networks* 10 (8) (1997) 1473–1494.
- [15] M. Fry, A. Ghosh, Application layer active network, *Comp. Nets.* 31 (7) (1999) 655–667.
- [16] S. Grossberg, Adaptive pattern classification and universal recoding. I. Parallel development and coding of neural feature detectors, *Biol. Cybern.* 23 (1976) 121–134.
- [17] S. Grossberg, Adaptive pattern classification and universal recoding. II. Feedback, expectation, olfaction, and illusions, *Biol. Cybern.* 23 (1976) 187–202.
- [18] T. Kohonen, Improved versions of learning vector quantization, *Proceedings of the IJCNN'90*, Vol. 1, 1990a, 545–550.
- [19] T. Kohonen, Self-organised formation of topologically correct feature maps, *Biol. Cybern.* 43 (1982) 53–69.
- [20] T. Kohonen, The self-organizing maps, *Proc. IEEE* 78 (9) (1990) 1464–1480.
- [21] S.W. Lee, D. Palmer-Brown, J. Tepper, C.M. Roadknight, Snap-drift: real-time performance-guided learning, *Proceedings of the IJCNN'03*, Vol. 2, 2003, pp. 1412–1416.
- [22] S.W. Lee, D. Palmer-Brown, J. Tepper, C.M. Roadknight, Performance-guided neural networks for rapidly self-organising active network management, in: A. Abraham, J. Ruiz-del-Solar, M. Koppen (Eds.), *Soft Computing Systems: Design, Management and Application*, IOS Press, Netherland, 2002, pp. 21–31.
- [23] C.P. Lim, R.F. Harrison, Modified fuzzy artmap approaches Bayes optimal classification rates: an empirical demonstration, *Neural Networks* 10 (4) (1997) 755–774.
- [24] G. MacLarty, M. Fry, Policy-based content delivery: an active network approach, *Proceedings of the IWCW'*, 2000.
- [25] I.W. Marshall, Application layer programmable internetwork environment, *BT Tech.* 17 (2) (1999) 82–94.
- [26] I.W. Marshall, et al., Active management of multi-service networks, *IEEE Proc. NOMS'2000*.
- [27] I.W. Marshall, Active network—making the next generation internet flexible, *BT Eng.* 18 (1999) 2–8.
- [28] I.W. Marshall, C.M. Roadknight, Provision of quality of service for active services, *Comp. Nets.* 36 (1) (2001) 75–85.
- [29] D. Palmer-Brown, An adaptive resonance classifier, Ph.D. Thesis, Department of Computer Science, University of Nottingham, 1991.
- [30] D. Palmer-Brown, High speed learning in a supervised, self growing net, in: I. Aleksander, I. Taylor (Eds.), *Proceedings of the ICANN'92*, Vol. 2, 1992, pp. 1159–1162.
- [31] D. Palmer-Brown, J.A. Tepper, H. Powell, Connectionist Natural Language Parsing, *Trends Cogn. Sci.* 6 (10) (2002) 437–442.
- [32] D. Tennenhouse, D. Wetherall, Towards an active network architecture, *Comp. Com. Rev.* 26 (2) (1996) 5–18.



robotics. His main research centres on neural networks, natural language processing and adaptive learning systems.

Sin Wee Lee was born in Melaka, Malaysia, in 1976. He graduated with first class honours in electronics and computing engineering from the Nottingham Trent University, Nottingham, UK, in 1999. His Ph.D. thesis focuses on the development of performance-guided neural network for active network management. From 2000 to 2001, he was a systems engineer at Malaysia Multimedia University in Malaysia. In December 2001, he joined the School of Computing, Leeds Metropolitan University, Leeds, UK, with a research scholarship from BT Research Laboratories in Neural Networks. As a research assistant, he works at the School of Computing, Leeds Metropolitan University with Dominic Palmer-Brown, on the improvement and development of connectionist language processing, improvements to the snap-drift algorithm and its applications, such as in wireless sensor networks and



and SRNs for thematic knowledge extraction and natural language processing. He was editor of the review journal *Trends in Cognitive Sciences* during 2000–2002 before rejoining Leeds Met.

Dominic Palmer-Brown is professor of neurocomputing and the leader of the Computational Intelligence Research Group, in the School of Computing at Leeds Metropolitan University, UK. The Group have active research links with several organisations, including British Telecom Research Labs, The Centre for Ecology and Hydrology, and with other universities. A key focus of our research is neurocomputing and related methods of adaptation and learning in cognitive science, intelligent data analysis, and pattern recognition. Dominic has published over 40 international conference and journal papers and supervised 10 Ph.D.s since 1993, having completed his own Ph.D. on an adaptive resonance classifier in 1991. His interests have principally concerned supervised and performance-guided ART, enhanced MLPs for intelligent data analysis, and architectures incorporating MLPs



Christopher Roadknight holds a B.Sc. in Biological Sciences and a M.Sc. in Computer Sciences from Manchester University and was awarded a Doctorate for his thesis on 'Transparent Neural Network Data Modelling' by Nottingham Trent University. He joined BT in 1997 and initially worked on characterising WWW requests and user behaviour, with applications to proxy cache performance modelling. His work within the programmable networks lab has focused on artificial-life based solutions for active network management.