Author(s): Lee, Sin Wee; Palmer-Brown, Dominic; Tepper, Jonathan; Roadknight, Christopher.
Article title: Snap-Drift: Real-time, Performance-guided Learning
Year of publication: 2003
Citation: Lee, S. W.; Palmer-Brown, D.; Tepper, J. A; Roadknight, C.M. (2003). "Snap-Drift: Real-time, Performance-guided Learning." In Proceedings of the International Joint Conference on Neural Networks (IJCNN'2003) (Portland, Oregon, 20th - 24th July), Vol. 2, pp. 1412–1416.
Link to published version: http://dx.doi.org/10.1109/IJCNN.2003.1223903
DOI: 10.1109/IJCNN.2003.1223903

Snap-Drift: Real-time, Performance-guided Learning

S. W. Lee, D. Palmer-Brown
Leeds Metropolitan University,
Computational Intelligence Research Group,
Beckett Park, LS6 3QS Leeds, UK.
http://www.lmu.ac.uk/ies/comp/research/cig/
J. A. Tepper
The Nottingham Trent University,
School of Computing and Mathematics,
Burton Street, NG1 4BU Nottingham, UK.
C. M. Roadknight
BTexact Technologies,
BTAdastral Park, Martlesham Heath, IP5 3RE Ipswich, UK

Abstract- **A novel approach for real-time learning and mapping of patterns using an external performance indicator is described. The learning makes use of the 'snap-drift' algorithm based on the concept of fast, convergent, minimalist learning (snap) when the overall network performance has been poor and slower, cautious learning (drift towards user request input patterns) when the performance has been good, in a non-stationary environment where new patterns are being introduced over time. Snap is based on Adaptive Resonance; and drift is based on Learning Vector Quantization (LVQ) [1]. The two are combined in a semi-supervised system that shifts its learning style whenever it receives a change in performance feedback. The learning is capable of rapidly relearning and restabilising, according to changes in feedback or patterns. We have used this algorithm in the design of a modular neural network system, known as Performance-guided Adaptive Resonance Theory (P-ART) [2,3]. Simulation results show that it discovers alternative solutions in response to a significantly changed situation, in terms of the input vectors (patterns) and/or of the environment, which may require the patterns to be treated differently over time.**

I. THE PERFORMANCE-GUIDED ART (P-ART) ARCHITECTURE

A. The P-ART system

The ART1 [7] network rapidly organises itself into a stable state due to fast learning, resulting in weights that no longer adapt. There is no external feedback to improve the performance of the network once it has stabilised.

The P-ART network proposed is a modular, multi-layered architecture as shown in Fig. 1. On the presentation of an input pattern at the input layer $F0_1$, 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 'snap-drift' algorithm. The matching and reset mechanism, however, is that of ART [11]; If no existing matching prototype is found, i.e. when the stored pattern prototypes are not a good match for the input, the winning $F2_1$ node is reset and another $F2_1$ node is selected, whose pattern prototype will then be matched against the input, and so on. 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 top three $F2_1$ nodes are used as the input for the sP-ART module for selecting an 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 points to (activates) pre-trained locations on the Kohonen Self-Organising Map (SOM) [10,12], which represent specific proxylets. If the proxylet is unavailable, one of its neighbours is selected (the most similar alternative available).

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* network decision. The measure must be used to encourage or discourage reselection of outputs (proxylet types) to occur in order to improve system performance.

B. The dP-ART Learning Principles

On the presentation of a binary input pattern I, the network attempts to categorise the input pattern by comparing it against the stored knowledge of the existing distributed output categories of $F2_1$ layer. This is achieved by calculating the bottom-up activation, using (1):

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

As this architecture is based on a distributed P-ART, there is more than one winning node, in this case D = 3. The three $F2_1$ nodes with the highest bottom-up activation are selected. If a distributed output category is found with the required matching level, using (2), as in ART:

This work is funded by British Telecom (BT) Research Laboratorics.

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

where vigilance parameter $0 < \rho < 1$, the three $F2_1$ nodes with the highest activation will enter into a resonant state and learn by modifying their weights to keep only the critical features for the selected output category.

The top-down learning of the network can be illustrated using (3):

$$w_{iJ}^{(new)} = (1 - p)(I \cap w_{iJ}^{(old)}) + p(w_{iJ}^{(old)} + \beta(I - w_{iJ}^{(old)})) \quad (3)$$

where

$w_{ij}^{(old)}$ = The top-down weights vectors at the start of the input presentation

p = Performance parameter

I = Binary input vectors

β = The 'drift' constant

In general, (3) can be stated as:

$$w_{iJ}^{(new)} = \alpha(\text{fast\_learning ART}) + \beta(\text{LVQ}) \quad (4)$$

where α-β balance is guided by performance feedback. So, in principles, although P-ART, like ART, is an unsupervised learning, but unlike ART, it is reinforced according to its performance. The network combines minimalist ART learning with Learning Vector Quantization (LVQ). By substituting p in (3) with 0 for poor performance, (3) can be simplified to:

$$w_{iJ}^{(new)} = (I \cap w_{iJ}^{(old)}) \quad (5)$$

Thus fast learning is invoked, causing the top-down weights to reach their new asymptote on each input presentation:

$$w_J \rightarrow I \cap w_J^{(old)} \quad (6)$$

In contrast, for excellent performance where $p = 1$, (3) can be simplified to:

$$w_{iJ}^{(new)} = (w_{iJ}^{(old)} + \beta(I - w_{iJ}^{(old)})) \quad (7)$$

Thus, a simple form of clustering or LVQ occurs at a speed determined by β.

It is assumed that there is a considerable interval between updates of p during which time new previously unseen requests are likely to appear. Equation (7), or indeed (3) whenever performance is not perfect, enables the top-down weights to drift towards the input patterns. With alternate episodes of $p = 0$ and $p = 1$, the characteristics of the learning of the network will be the joint effects of the (5) and (6). This joint effect can enable the network to learn using fast and convergent, snap learning when the performance is poor, yet be able to drift towards the input patterns when the performance is good.

If no existing matching prototype is found, i.e. when the stored w_J does not match the input, then the winning $F2_1$ node is reset and another $F2_1$ node with the highest T_i is selected, whose prototype will be compared against the input vector, and so on. 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 and an associated set of weights. This new output node is then associated with the current input vector by making its weight vector equal to the input vector.

The bottom-up learning of the P-ART can be illustrated using the following:

$$w_{jI}^{(new)} = (1-p)\frac{I \cap w_{jI}^{(old)}}{|I \cap w_{jI}^{(old)}|} + p(w_{jI}^{(old)} + \beta\frac{I - w_{jI}^{(old)}}{|I - w_{jI}^{(old)}|}) \quad (8)$$

where

$w_{ji}^{(old)}$ = The bottom-up weights of the network at the start of the input presentation.

At the beginning of the first input presentation, the bottom-up weight w_{ji} are assigned with initial values corresponding to the initial top-down weights w_{ij} values using (9):

$$w_{ji}(0) = \frac{w_{ij}(0)}{1 + N} \quad (9)$$

where N = Number of input nodes

By selecting this small initial value of w_{ji}, the network is 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 (8), the bottom-up learning of the network can be illustrated as follows:

$$w_{jI}^{(new)} = \frac{I \cap w_{jI}^{(old)}}{|I \cap w_{jI}^{(old)}|} \quad (10)$$

This is fast, convergent learning.

In contrast, if the network encountered perfect performance after a considerable interval, (8) can be simplified as follows:

$$w_{jI}^{(new)} = w_{jI}^{(old)} + \beta\frac{I - w_{jI}^{(old)}}{|I - w_{jI}^{(old)}|} \quad (11)$$

This will resulted in the weights drifting towards the input vector.

Essentially, the principle is that drift, by itself, will only results in slow (depending on β) reselection over time, thus keeping the network up-to-date without a radical set of reselections for exiting patterns. By contrast, snapping results in rapid reselection of a proportion of patterns to quickly respond to a significantly changed situation, in terms of the

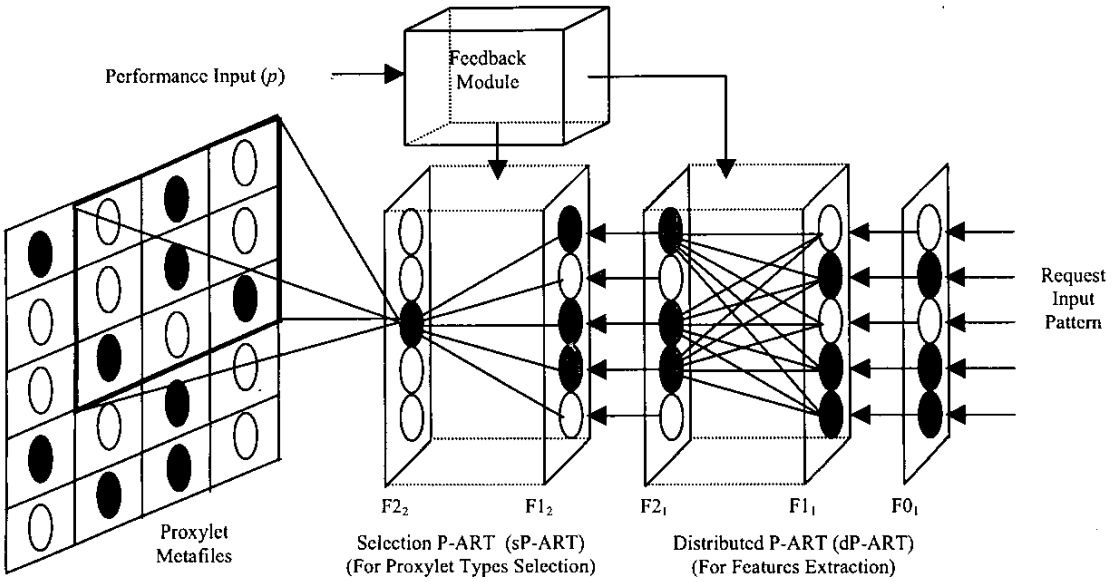

Fig. 1 Architecture of the P-ART network

input vectors (requests) and/or of the environment which may require the same requests to be treated differently by snapping from a new position in weight space. Thus, new category node selection may occur for one of two reasons: as a result of the drift itself, or as a result of the drift enabling a further snap to occur (since drift has moved away from convergence) if performance p goes down.

C. The sP-ART Learning

The distributed output representation of categories produced by the dP-ART acts as input to the sP-ART. The architecture of the sP-ART is the same as that described in section I with one exception; only the $F2_2$ node with the highest activation will be selected for learning. Each output node of the sP-ART represents the set of available proxylet types in the network. These proxylet types have also 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' onto 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 wired to the appropriate region of the SOM. The effect of learning and relearning within the sP-ART module is that specific output nodes will represent different groups of input patterns until the performance feedback indicates that it is indexing the correct regions of the SOM and thus selecting the correct proxylets.

D. The Performance Feedback

The external performance feedback into the P-ART will reflect the performance requirement in different circumstances. Various performance feedbacks profiles in the range {0,1} are fed into the network to evaluate the dynamic stability and performance responsivity of the learning. Initially, some very basic tests with performances of 1 or 0 were evaluated in a simplified system [2,3]. Below, the simulations involve computing the performance based on a parameter associated with the winning output neuron. Ultimately, a realistic commercial external performance feedback criteria will be established, which will be obtained from BT, to evaluate the improvement in performance of the network learning under realistic external performance feedback. In the BT application, functions which contribute to good / poor performance include latencies for 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.

II. THE BRITISH TELECOM (BT) APPLICATION

A. Application Layer Active Network (ALAN)

The ALAN architecture was first proposed by Fry and Ghosh [4] 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 dealing with the current system architecture and equipment. This approach relies on the redirecting of selected request packets into the EEP, where the appropriate proxylets can be executed to modify the packets contents without impacting

on the router's performance and thus does not need any additional standardization.

B. Automated Active Network Management using Distributed Genetic Algorithms

Recently, a novel adaptive approach using Distributed Genetic Algorithm (GA) to this problem of automated network management was introduced by Marshall and Roadknight from British Telecom (BT) Research Laboratories, which solves this problem with some success.

The algorithm was applied to the adaptive management solution and to the differentiated quality of service mechanism defined by Marshall & Roadknight [5,6] for ALAN and has shown promising results with respect to increasing the performance of the ALAN network.

P-ART is used as a means of finding and optimising a set of conditions that produce optimum proxylets selection in the Execution Environment for Proxylets (EEP), which contains all the frequently requested proxylets (services).

III. SIMULATIONS

A. Assessment and Evaluation of Results

This section presents the simulations and evaluation of results performed on the P-ART module and thus evaluates the behaviour of the 'snap-drift' algorithm.

The test patterns consist of 100 input vectors. Each test pattern characterizes the features/properties of a realistic network request, such as bandwidth, time, file size, loss and completion guarantee. These test patterns were presented in random order for 25 epochs where the performance, p, is calculated according to the average bandwidth of selections. This on-line continuous random presentation of test patterns simulates the possible real world scenario where the order of patterns presented is random so that a given network request might be repeatedly encountered while others are not used at all.

B. Results

In Fig. 2, 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 can be any length.

Fig. 3 shows the selection frequency of the proxylet type. In this case, we have the following bandwidth bands:

- Low bandwidth proxylet: $0 \rightarrow 600$ Kb/s
- Median bandwidth proxylet type: $601 \rightarrow 1200$ Kb/s
- High bandwidth proxylet type: >1201 Kb/s.

At the first epoch (refer to Fig. 2), the performance is set to 0 to invoked 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, from Fig. 2, there is a significant decrease in performance. As illustrated in Fig. 3, this is due to 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 28^{th} 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 new input patterns are further introduced or there is a change in the performance circumstances, the network will maintains at this high level of performance. On different run, as shown in Fig. 4, 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:

- Short execution time proxylet: $1 \rightarrow 300$ ms
- Median execution time proxylet type: $301 \rightarrow 600$ ms
- Long execution time proxylet type: > 600 ms

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

These results indicate that the performance could be modified using a range or combination of performance parameters. Other parameters such as cost, file size will be added to the performance calculation to produce a more realistic simulation of network circumstances in the future.

IV. CONCLUSIONS

A neural network architecture 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 network performance, based on the external performance feedback. These system properties have been confirmed by the results obtained from the experiments performed using the P-ART module, which was evaluated using performance feedback scenarios.

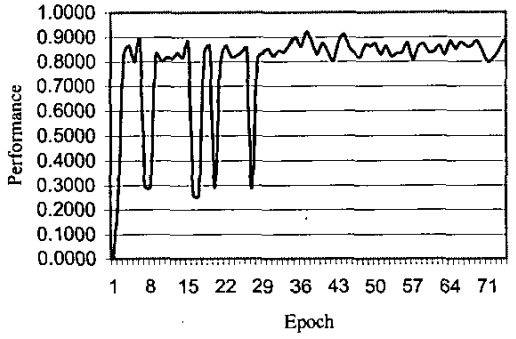

Fig 2 Performance levels of the network

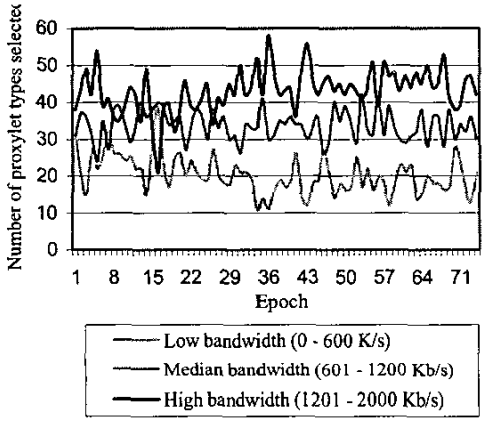

Fig. 3 Selection frequency of the 3-bandwidth bands proxylet types at each epoch.

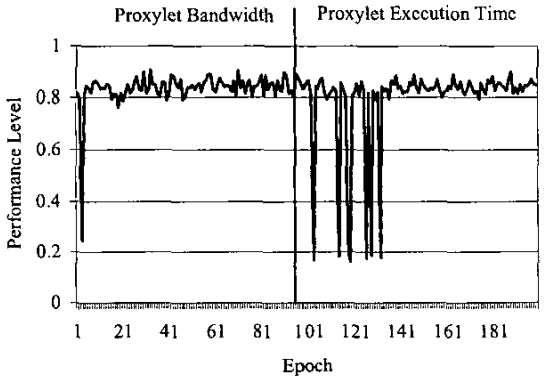

Fig. 4 Performance level of the P-ART system with alternate episode of performance criterion.

REFERENCES

[1] T. Kohonen, "Improved Versions of Learning Vector Quantization", *International Joint Conference on Neural Networks*, San Diego, vol I, 545 – 550, 1990.

[2] S.W. Lee, D. Palmer-Brown, J. Tepper and C.M. Roadknight, "Performance-guided Neural Networks for Rapidly Self-Organising Active Network Management", in: *Soft Computing Systems: Design, Management and Application*, A. Abraham, J. Ruiz-del-Solar, and M. Koppen, Eds., Netherland: IOS Press, 2002, pp. 21 - 31.

[3] S.W. Lee, D. Palmer-Brown, J. Tepper and C.M. Roadknight, "Performance-guided Neural Network for Self-Organising Network Management", *Proceeding of London Communications Symposium*, University College London, pp. 269 – 272, Sep 2002.

[4] M. Fry and A. Ghosh, "Application Layer Active Network", *Computer Networks*, vol. 31(7), pp. 655 – 667, 1999.

[5] I.W. Marshall and C.M. Roadknight, "Provision of Quality of Service for Active Services", *Computer Networks*, vol. 36(1), pp. 75 – 85, 2001.

[6] I.W. Marshall and C.M. Roadknight, "Differentiated Quality of Service in Application Layer Active Networks", in: *Active Networks, LNCS 1942*, Yasuda, Eds., Springer-Verlag, 2000, pp. 358-371.

[7] G.A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organising Neural Pattern Recognition Machine", *Computer Vision, Graphics and Image Processing*, vol. 37, pp. 54-115, 1987.

[8] D. Palmer-Brown, "High Speed Learning in a Supervised, Self Growing Net", in: *Proceeding of ICANN 92*, I. Aleksander and I. Taylor, Eds., Brighton, vol. 2, pp. 1159-1162, 1992.

[9] S. Barker, H. Powell and D. Palmer-Brown, "Size Invariant Attention Focusing (with ANNs)", *Proceeding of International Symposium on Multi-Technology Information Processing*, 1996.

[10] T. Kohonen, "The Self-Organizing Maps", *Proceeding of the IEEE*, vol. 78(9), pp. 1464 –1480, 1990.

[11] G.A. Carpenter, S. Grossberg, "The ART of Adaptive Pattern Recognition by a Self-Organising Neural Networks", *IEEE Computer*, vol. 21(3), pp. 77 – 88, 1988.

[12] T. Kohonen, "Self-organized Formation of Topologically Correct Feature Maps", *Biological Cybernetics*, vol. 43, pp. 53 – 69. Reprinted in Anderson and Rosenfeld, pp. 511-521, 1998.