FAST LEARNING NEURAL NETS WITH ADAPTIVE LEARNING STYLES

DOMINIC PALMER-BROWN, SIN WEE LEE, JON TEPPER and CHRIS ROADKNIGHT

Computational Intelligence Research Group,
School of Computing, Leeds Metropolitan University,
Beckett Park, Leeds LS6 3QS (d.palmer-brown@lmu.ac.uk).

Abstract - There are many learning methods in artificial neural networks. Depending on the application, one learning or weight update rule may be more suitable than another, but the choice is not always clear-cut, despite some fundamental constraints, such as whether the learning is supervised or unsupervised. This paper addresses the learning style selection problem by proposing an adaptive learning style. Initially, some observations concerning the nature of adaptation and learning are discussed in the context of the underlying motivations for the research, and this paves the way for the description of an example system. The approach harnesses the complementary strengths of two forms of learning which are dynamically combined in a rapid form of adaptation that balances minimalist pattern intersection learning with Learning Vector Quantization. Both methods are unsupervised, but the balance between the two is determined by a performance feedback parameter. The result is a data-driven system that shifts between alternative solutions to pattern classification problems rapidly when performance is poor, whilst adjusting to new data slowly, and residing in the vicinity of a solution when performance is good.

***Keywords*:** neural networks, fast learning, performance feedback, adaptive learning styles.

1. MOTIVATIONS AND OBJECTIVES

There are some basic observations and principles that motivate research into neural networks and other systems that are capable of leaning 'on the fly'. These concern the ability to rapidly adapt to discover provisional solutions that meet criteria imposed by a changing environment.

1. 1 Provisional Learning

The adaptive systems of interest in this type of research are not required to solve an optimisation problem in the traditional sense; they search heuristically for good solutions (solutions that are fit for purpose according to the chosen criteria of the target application) in a hyperspace that may contain many plausible solutions. However, heuristic information may be expressed by an objective function of some kind, which the system tries to 'optimise'. The classic example is error minimisation, in which in general the data is imperfect, e.g. limited, sparse, missing, error-prone, and subject to change (non-stationary). Therefore, the error minimum is really just a local minimum: local to a subset of data and an episode of time.

Whilst this does not preclude the discovery of solutions that work for all data-time, it does mean that such generalisation involves extrapolations and assumptions that cannot be justified on the sole basis of the available information. In such circumstances, it is reasonable, when a new candidate solution is found, for it to be held – as a provisional hypothesis – until or unless it is rejected, or until it can be replaced by a stronger hypothesis.

1.2 Fast Learning

Slow, iterative and intensive sampling based methods (eg. Gradient descent methods, and Bayesian methods involving Monte Carlo and related methods) are *inherently* non-real-time, in the sense that they require multiple presentations of sets of patterns or samples, and therefore they cannot respond to the changing environment *as it is* changing. This contrasts sharply with the human case. Humans learn 'as they go along', to a significant extent, without the need for multiple presentations of each exemplar or pattern of information.

1.3 Performance-guided Learning

An important concern in artificial intelligence is how to combine top-down and bottom-up information. This applies to learning systems. For example, reinforcement learning is very effective at rewarding successful strategies, or moves, during learning; supervised learning is a powerful means of modifying an ANN when it makes mistakes; and genetic algorithms are effective at selecting for improvement across generations of solutions. These are important and effective approaches, not to be dismissed simply because they are not fast, or because they are computationally intensive.

 ISBN 3-936150-25-7

Fascinating results and innovations are still occurring with these approaches, as this conference testifies [Vieira et al 2003, Andrews 2003, Lee et al 2003]. Equally unsupervised learning, which does not harness top-down information, is an extremely useful tool, for example as an alternative or complement to clustering; but in its purest form it does not (by definition) make use of any information on the current performance of learning, in order to guide adaptation in appropriate directions.

Ideally, learning should be rapid, and yet capable of taking external indicators of performance into account; and it should be capable of reconciling the data (bottom-up) with feedback concerning how the ANN is organising the data (top-down).

2. ADAPTIVE RESONANCE

The points raised above have led to the development of PART (Performance-guided Adaptive Resonance Theory), which has two of antecedents, ART (the original Adaptive Resonance Theory), and SMART (Supervised Match-seeking ART).

2.1 Adaptive Resonance Theory (ART)

ART [Carpenter and Grossberg, 1988] performs unsupervised learning. A winning node is accepted for adaptation if:

$\frac{|w \cap I|}{|I|} \geq \rho$, where w is the weight vector, I is the input vector and ρ is the so-called vigilance parameter, which therefore determines the level of match between the input and the weights required for a win. Weight adaptation is governed by: $w_{iJ}^{(new)} = n(I \cap w_{iJ}^{(old)}) + (1\text{-}n)\ (w_{iJ}^{(old)})$. As a result, only those elements present in both I and w remain after each adaptation, and learning is fast. In fact, it is guaranteed to converge in 3 passes of any set of patterns when n=1.

2.2 Supervised Match-seeking Adaptive Resonance Tree (SMART)

In order to convert ART into a supervised learning system that would therefore learn prescribed problems, SMART was developed [Palmer-Brown, 1992]. In this case the winning nodes are labelled with a classification. When a node with a label wins, if the classification is correct, learning proceeds as usual. If the class is wrong, a new node is initialised with the values of I, so that it would win in competition with the current winning node. An upper limit may be imposed on the number of nodes, in which case further learning results in some nodes becoming pointers to subnets, which learn in the same way as the first net. Hence the system is a fast, self-growing network tree.

2.3 Information Loss

The main limitation that was found with ART and SMART was the 'one strike and you're out' nature of the adaptation. Nodes sometimes need to retain information that is relevant to only a subset of the patterns for which they win. The $w \cap I$ intersection is responsible for this information loss, but it is also the reason for the rapidity and stability of the learning process. Thus, the challenge is to retain these positive characteristics whilst preventing the learning from throwing away information when it is needed. This objective, along with the points made in section 1, has led to the development of Performance-guided Adaptive Resonance (PART).

3. PERFORMANCE-GUIDED ADAPTIVE RESONANCE (PART)

A non-specific performance measure is used with PART because, in many applications, there are no specific performance measures (or external feedback) available in response to each *individual* network decision. PART consists of a distributed network and a non-distributed network, in order to perform feature(s) extraction followed by feature classification, in two stages. Fig. 1 illustrates the architecture in the context of a particular application [Sin Wee et al, 2002].

3.1 dP-ART Learning

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

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

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

$$\frac{|wI \cap I|}{|I|} \geq \rho \qquad (2),$$

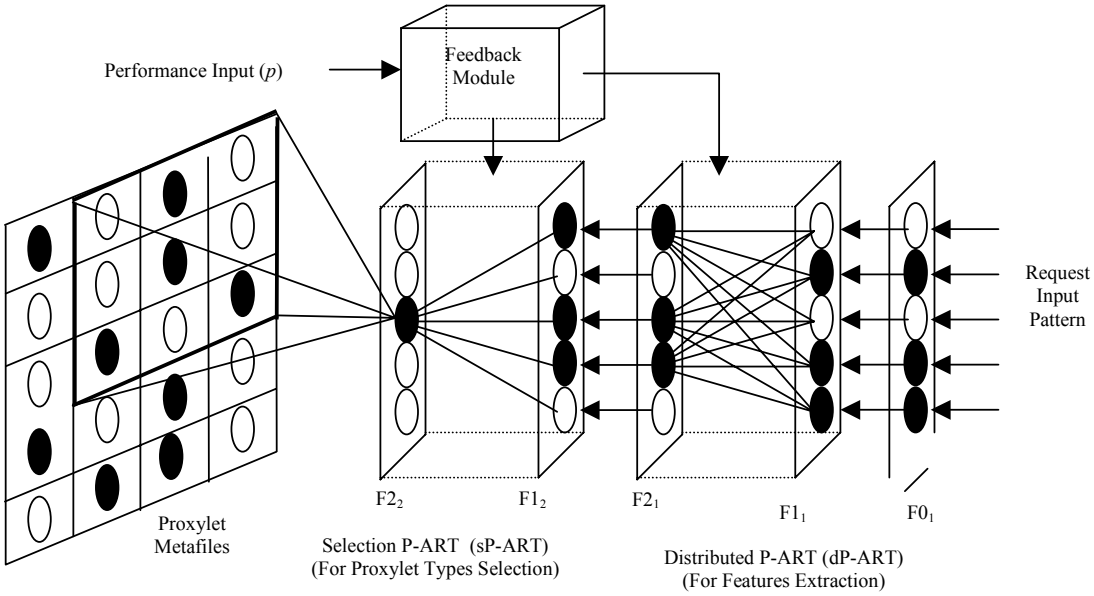

Figure 1. The PART System

Then learning occurs, according to equation (3):

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

where $w_{ij}^{(old)}$= the top-down (a similar equation applies for the bottom-up weights) vectors at the start of the input presentation; p = performance parameter; I= binary input vector; and β = the 'drift' constant. The effect of (3) is $w_{iJ}^{(new)} = \alpha$ (fast_ART learning) + β (LVQ) (equation 4), where LVQ stands for Learning vector quantization.

The α-β balance is determined by performance feedback. Therefore P-ART does unsupervised learning, but its learning style is determined by its performance, which may be updated at any time. So PART combines minimalist ART learning with Learning Vector Quantization (LVQ) [Kohonen, 1990], and by substituting p in (3) with 0 for poor performance, (3) can be simplified to:

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

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

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

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

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

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

Equation (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 enables 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. Drift will only result in slow (depending on β) reclassification of inputs over time, keeping the network up-to-date, without a radical set of reclassifications 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 input vectors (requests) and/or of the environment, which may require the same requests to be treated differently. Thus, a new classification 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, once the drift has moved weights away from convergence.

3.2 sP-ART

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 above except that only the $F2_2$ node with the highest activation is selected for learning. The effect of learning within sP-ART and

dP-ART is that specific output nodes will represent different groups of input patterns until the performance feedback indicates that sP-ART is indexing the correct outputs (called proxylets in the target application).

4 AN EXAMPLE APPLICATION

4.1 The Performance Feedback

The external performance feedback into the P-ART reflects the performance requirement in different circumstances. Various performance feedbacks profiles in the range {0,1} are fed into the network to evaluate the dynamics, stability and performance responsivity of the learning. Initially, some very basic tests with performances of 1 or 0 were evaluated in a simplified system [Sin Wee et al, 2002]. Below, the simulations involve computing the performance based on a parameter associated with the winning output neuron. In the target application, provided by BT [Marshall and Roadknight, 2000, 2001], factors which contribute to good/poor performance include latencies for proxylet (eg software) requests with differing time to live, dropping rate for request with differing time to live, and different charging levels according to quality of service, and so on.

4.2 Application Layer Active Network (ALAN)

The ALAN architecture was first proposed by [Fry and Ghosh, 1999] to enable users 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 packet's contents without impacting on the router's performance.

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

4.3 Simulations

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 first presented in random order for 25 epochs where the performance, p, is calculated according to the average bandwidth of selections. This continuous random-order presentation of test patterns simulates the real world scenario where the order of patterns presented is such that a given network request might be repeatedly encountered, while others are not used at all.

4.4 Results of simulations

In Figure 2, we show the performance calculated across the simulation epochs. An epoch consists of 50 patterns, randomly selected. Performance feedback is updated at the end of each epoch. 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 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, 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 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 (reclassifications), resulting in improved performance.

By the 28^{th} epoch, where $p = 0.81$, the performance has stabilised around the average performance of 0.85. At this stage, most of the possible input patterns have been encountered several times. Until new input patterns are introduced or there is a change in the performance circumstances, the network will maintain at this high level of performance.

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 → 300 ms; Median execution time proxylet type: 301 → 600 ms; Long execution time proxylet type: > 600 ms.

This criterion is fed into the P-ART at every 100th epoch. The result indicated when the new performance criterion is introduced in the 100th epoch, rapid reselection of a proportion of the patterns occurs on a consistent basis.

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.

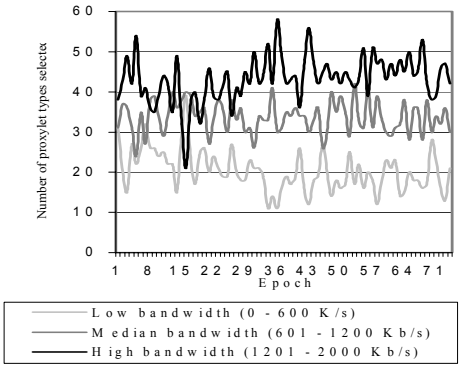

Fig 2 Performance levels of the network
Figure 2. Performance.

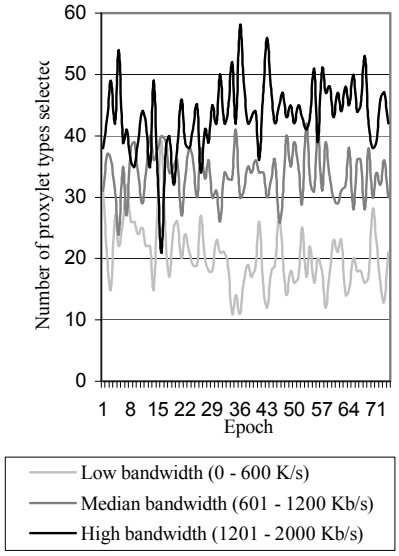

Fig. 3 Selection frequency of the 3-bandwidth bands of proxylet types at each epoch.

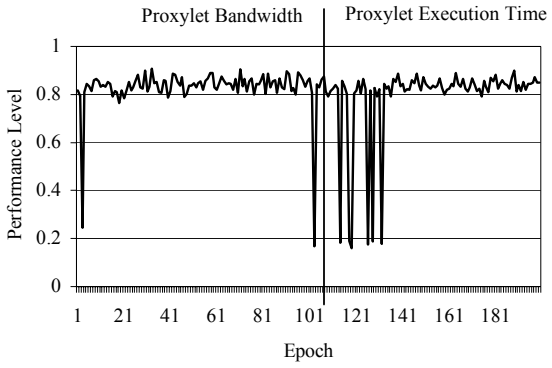

Fig. 4. Performance level before/after a problem change.

5. CONCLUSION

The PART system is able to adapt rapidly to changing circumstances. It manages to reconcile top-down and bottom-up information by finding a new provisional solution to the pattern classification problem whenever performance deteriorates. There is clearly potential to apply this approach to a wide range of problems, and to develop it in order to fully explore the objectives stated in section 1.

REFERENCES

S. G Andrews. “Novel neural network methods for describing attributes contained within lesions images”. In Proc. of ESM2003.

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.

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

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

J Lee, S Mian, R Rees, and G Ball. “Preliminary Artificial Neural Network Analysis of SELDI Mass Spectrometry Data for the Classification of Melanoma Tissue”. In Proc. of ESM2003.

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.

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.

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

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

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.

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.

C. Vieira, P Mather, P Alpin. "Improving Artificial Neural Network Performance by Using Temporal-Spectral Features for Agricultural Crop Classification". In Proc. of ESM2003.