Learning Complex Combinations of Operations in a Hybrid Architecture

L. Andrew Coward and Tamás D. Gedeon
Department of Computer Science
Australian National University
Canberra, ACT 0200, Australia
E-mail: andrewc@cs.anu.edu.au and
tom.gedeon@anu.edu.au

Uditha Ratnayake
Dept. of Electrical and Computer Engineering
The Open University of Sri Lanka, PO BOX 21
Nawala, Nugegoda, Sri Lanka
E-mail: udithaw@ou.ac.lk

Abstract. The reasons why machine learning appears limited to relatively simple control problems are analyzed. A primary issue is that any condition detected by a learning system acquires multiple behavioural meanings. As learning continues, the need to preserve these meanings severely constrains the architectural form of the system. A hybrid architecture called the recommendation architecture in which the preservation of such meanings is explicitly managed is compared with a wide range of alternative learning approaches. It is concluded that systems with this recommendation architecture have the capability to learn to solve complex control problems.

I. INTRODUCTION

Machine learning is currently limited to much less operationally complex problems than the types of problem which can be solved by design in conventional electronic systems. The limitations to current approaches are associated with the stability-plasticity dilemma, and this dilemma can also be expressed as the difficulty of maintaining adequate operational meanings for device outputs throughout learning. The type of device algorithm is a major limiting factor to the ability to maintain such adequate meanings. A different type of device algorithm and an architectural framework called the recommendation architecture within which such algorithms can be used is described. This framework is compared with a range of other approaches including self organizing maps, adaptive resonance theory, the use of input pseudopatterns to refresh prior learning, and fuzzy logic. The conclusion is reached that the recommendation architecture approach can limit the interference between later and prior learning, and has strong potential for learning to solve operationally complex problems.

II. OPERATIONAL SYSTEMS

An operational system can be defined as one which acts upon itself and its environment in order to achieve specified target states, or objectives, for itself and its environment. Such a system detects conditions in itself and its environment and associates those conditions with behaviours which will result in target states. An operationally complex system is one which has a large input space, large numbers of sometimes conflicting objectives, large numbers of possible behaviours, many different conditions indicating each behaviour where conditions may include information from input states at different times, and which requires sequences and/or combinations of behaviours to achieve objectives.

In principle, such a system could be implemented as an immense look-up table, with each operationally relevant condition of the total input state or sequence of such states listed separately with its corresponding behaviour. Such an implementation would require impractical levels of information recording and processing resources. In practice, therefore, systems must detect a much smaller population of conditions within different subsets of their input spaces, and associate many different combinations of this limited set of conditions with different behaviours. Identifying an appropriate but limited set of such conditions is a primary design problem. However, solving the resource problem in this fashion introduces another problem. If the operation of the system must be changed without excessive proliferation of the conditions which must be detected, in general some previously defined conditions must be modified. Such modifications will tend to introduce undesirable side effects into other operations which employ the original unmodified conditions. It will therefore be necessary to find a compromise between use of resources and ease of modification. Coward [7] has argued that this compromise requires that the system be organized as a modular hierarchy. Each module detects a portfolio of conditions, and outputs from a module limit the range of appropriate behaviours to a relatively small subset. To make change practicable, modules are defined in such a way that conditions detected by one module are used as little as possible by other modules: information exchange between modules is minimized as far as possible.

Most approaches to machine learning, such as artificial neural networks (ANNs), fuzzy logic, reinforcement learning and various combinations of these approaches have only been able to solve relatively simple operational problems. For example, pattern recognition problems are operationally simple in having fairly small numbers of alternative behaviours (i.e. the different categories to be identified) and no conflicting objectives. Control problems tend to have relatively small input spaces and few possible behaviours, or to have different behaviour types which are fully determined by independent input spaces, or to have little learning after an initial training period. A major issue with, for example,

ANNs, as the complexity of the task to be learned increases is interference between later learning and earlier learning. This interference can result in essentially complete destruction of earlier learning and is known as the sensitivity-stability problem [12], the stability-plasticity problem [3] or the catastrophic forgetting problem [9]. There is an interesting analogy between this issue and the issue of practical operational change which forces complex electronic systems to adopt modular hierarchies.

There have been attempts to define modular hierarchies of ANNs for more complex problems. One approach uses modules trained to detect features and objects made up of groups of features. Such approaches can be effective for pattern recognition but have not been applied to learning operational complexity. Another approach is based on networks of networks in which the meaning of information generated by individual networks is undefined, and coordination between networks is dependent upon a process analogous with convergence on attractor states in physical dynamical systems [e.g. 2]. However, how dynamical systems would support the information processing required to manage behaviour is not clear [8].

Another attempt to define an ANN modular hierarchy is the expert architecture developed by Jordan and Jacobs [13]. In this approach it is assumed that the system input space can be separated into segments, and appropriate supervised learning can occur in independent expert networks each receiving inputs from one segment. A gating network can then determine system behaviour by taking a weighted average of all the network outputs. However, the assumptions made, including the existence of a clean separation of the input space into segments and the absence of information exchange between different networks, means that the solution to the target problem is either operationally simple or can employ unlimited resources. The approach has only been applied to relatively simple operational problems [e.g. 1].

III. The Learning Issue

An operationally complex system must detect conditions within its input states and associate different combinations of those conditions with different behaviours. A system which heuristically defines its own operations must define and modify the conditions it detects and/or the associations between conditions and behaviours. The greater the complexity of the operations which must be heuristically defined, the greater the degree of modification to conditions and associations which must occur in the course of learning. However, if conditions and associations can be substantially specified a priori from external design knowledge, the difficulty of the learning problem is reduced.

For example, in fuzzy logic [14], the system inputs available to guide behaviour are defined a priori. Inputs are behaviourally significant indicators and not just raw measurements derived from large numbers of elementary sensors. Fuzzy variables are defined a priori as degrees to which specified combinations of system inputs are present. Rules associating combinations of fuzzy outputs with behaviours are learned, generally using domain expert guidance, and learning is limited to operations within a predefined domain. Such systems have no capability to define new input variables.

In a system which learns a complex combination of operations with limited a priori knowledge, there must be some means to ensure that the conditions are adequate to distinguish between behaviourally significant differences in input states. It will therefore be essential to have the capability to define new conditions. The fundamental issue is then that once an association between a group of conditions and a behaviour has been defined, any change to one of the conditions in the group may reduce the validity of the association. In an operationally complex system, detection of a condition will acquire multiple meanings, and such meanings must be adequately preserved. The critical problem is therefore the type and degree of changes to conditions which can be consistent with preservation of adequate operational meanings.

Statistical clustering attempts to organize unlabeled input vectors into clusters or "natural groups" in such a way that vectors within a cluster are "more similar" to each other than to vectors assigned to other clusters [11]. In clustering there may be no a priori information indicating natural groups or similarity measures. Such clustering is therefore one approach to heuristically defining operationally useful conditions, and various statistical clustering algorithms are instantiated by different unsupervised ANNs. To divide vectors into groups which possess strong internal similarities, a criterion function is used and a grouping is sought which optimizes the value of the function (i.e. maximizes or minimizes). For example, self organizing maps [15] assign each member of a large population of possible input vectors to one or more of a fixed small number of outputs. Assignment is on the basis of the similarity between the input vector and the reference vector for each output. An input is assigned to the output with the greatest similarity, and the reference vector for that output is changed slightly in the direction of the input.

Such ANNs thus have the capability to heuristically define conditions within large input populations based on information similarity, and associate these conditions with behaviours (generally useful categories). However, as mentioned earlier, ANNs suffer from interference between later learning and earlier learning which limits their capability to learn complex combinations of operations. To understand the source of this issue and how it might be overcome, consider first the nature of the conditions detected at the device level.

The output of a device in response to the presence of a specific combination of active inputs indicates that the condition specified by that combination of active inputs is present. Networks synthesize the outputs of many devices into a network output indicating the presence of some more operationally significant condition. The network output is itself the outputs(s) of one or more devices assigned to provide that output. In an operationally complex system a device output could acquire a wide range of different

meanings. The alternative, that no condition have multiple operational meanings, would require an impractical level of resources.

There are two aspects to the problem of retaining meaning as the conditions detected by a device change. One is whether a repetition of a condition which generated an output in the past will again generate an output. The other is the degree of similarity between a condition which generated an output in the past and different conditions which subsequently generate the same output.

In ANNs, the condition detected by a device is defined by the weights assigned to its inputs. These weights are evolved independently. In supervised learning, such evolution is on the basis of feedback of the consistency between network output in response to an input and some known target output. In unsupervised learning, evolution is on the basis of some criterion of similarity between the network input and network prototype inputs. A device will not necessarily produce an output in response to an exact repetition of an earlier input condition, and the relationship between different conditions generating an output from the same device is complex. The device algorithm is thus at the root of the catastrophic forgetting problem.

Several approaches have been proposed to overcome this problem. One approach is to constantly repeat subsets of earlier training during later learning [e.g. 19]. Robins [20] introduced the concept of repeating pseudo-inputs or approximations of the original inputs. However, as operational complexity increases, achieving an adequate balance between old and prior learning will be increasingly difficult. Another approach is to place additional constraints on the degree to which the conditions programmed on devices can change. An extensively investigated example of this approach is adaptive resonance theory (ART) [3]. In ART, as in a self-organizing map, an input vector is assigned to an output condition if it is sufficiently similar to the reference vector for that condition, and such an assignment results in a modification of the reference vector in the direction of the input. In contrast with regular self organizing maps, the number of output conditions is not predefined, and if an input vector is not sufficiently similar to any existing output vector, a new output is defined with the input as its initial reference vector. Although this approach reduces the degree of change, an exact repetition of an input condition will still not necessarily generate an output.

A more fundamental approach to retaining meaning during heuristic definition of conditions in neural networks was proposed by Coward [4]. In this approach, device algorithms are defined in such a way that once a device has generated an output in response to an input condition, it will always in the future generate an output in response to an exact repetition of that condition. Such a device also requires inputs defining when it will first respond to an input condition. These inputs are derived from specific groups of other devices as discussed below, but such inputs do not form parts of conditions. A device implementing a simple version of this type of algorithm is illustrated in figure 1.

The portfolio of conditions to which a device responds can expand but not contract. Because a higher level module is made up of a group of these portfolio definition devices, and the outputs of such a module are the outputs of a subset of the devices in the group, module portfolios also expand but do not contract. A condition can be added to a device portfolio if it meets a number of similarity criteria defined at the device level and at the level of various groups of devices making up higher level modules. These criteria can be managed to ensure that an adequate meaning is retained as device portfolios expand. As in ART, new portfolios can be added if a condition does not correspond closely enough with any existing portfolio, but in contrast with ART, the addition of new portfolios can be initiated by indications that the operational discrimination of the existing portfolios is inadequate, portfolios can be added on different modular levels, and once a portfolio has detected a specific condition it will detect any exact repetition of the condition.

Because condition portfolios can expand but not contract, portfolio definition device outputs cannot be used directly to control behaviour. For example, negative consequences following a behaviour cannot reduce a portfolio. The associations between conditions and behaviours must therefore be established in a separate subsystem, using devices in which relative input weights are adjusted. These devices can use supervised or reinforcement learning to associate different combinations of the defined conditions with different behaviours.

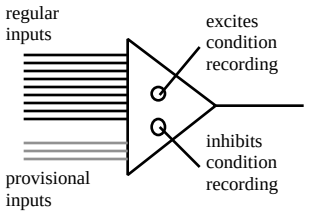

Fig. 1. A simple condition recording device. The device produces an output in response to any high proportion of its regular inputs. Active provisional inputs are added to the regular set if inputs which excite recording are active, inputs which inhibit recording are inactive, regular inputs are active to a level just below that required for the device to produce an output, and the total of active regular plus active provisional is sufficient to produce output.

Since the conditions detected are not directly influenced by behavioural feedback, they will not correlate unambiguously with behavioural conditions like features or categories. The presence of a condition can therefore never be interpreted as a command for a particular behaviour or recognition of a particular category, but only as a recommendation for a group of possible behaviours or categories. The function of the separate subsystem can therefore be interpreted as finding the strongest recommendation. The organization of portfolio definition devices in a modular hierarchy is called clustering because it clusters inputs into conditions, the separate subsystem is called competition because it resolves competitions between

alternative behavioural recommendations, and the overall system form is called the recommendation architecture. Because condition portfolios on every level expand but do not contract, and the definition of conditions is separated from the creation of associations between conditions and behaviours, this recommendation architecture is in principle less susceptible to interference between later learning and earlier learning.

IV. Implementations of Systems with the Recommendation Architecture

Electronic implementations of recommendation architecture systems have been investigated for several learning problems including telecommunications network management, document classification, and modeling human memory [6; 7; 10; 18]. The implementation used to investigate the effects of later learning on prior learning uses the condition recording device illustrated in figure 1. These devices are arranged in four layers, with columns across the layers organized into arrays. Two columns of an array are illustrated in figure 2. These four layers make up the clustering modular hierarchy. The levels in this hierarchy are device, small area of devices on one layer, and sequence of corresponding areas making up a column. Outputs from clustering go to the competition subsystem, with one device in the layer corresponding with each possible behaviour as in figure 2.

Initially, a portfolio recording device has a randomly selected group of provisional condition defining inputs. All these inputs have the same weight. Provisional inputs cannot cause a device to produce an output in the absence of appropriate change management inputs. If appropriate change management inputs are present, and if a sufficiently large subset of the provisional inputs are active, this currently active subset is selected as the initial condition in the device portfolio and the device produces a binary output. The active inputs become regular, in the sense that they will result in an output independent of change management inputs if the number of active inputs exceeds a threshold. The threshold of the device is set at or slightly below the size of the active subset. The device at this point will therefore produce an output in response to any large subset of original active set. Subsequent changes can reduce the threshold and/or add small proportions of additional regular inputs from a provisional group. The equality of all weights of inputs which define conditions to which a device will respond means that in a sense the portfolio recording device is binary or "weightless". However, weightless neural algorithms [17] do not ensure that once a device has produced an output in response to an input condition it will always generate an output in response to an exact repetition of that condition.

Condition defining inputs to layer 1 devices in a column are system inputs, those to layer 2 are layer 1 device outputs and so on. The complexity of conditions detected thus increases through the layers, where complexity is defined as the total number of system inputs which contribute to a condition. There is a single special purpose layer 4 device associated with the column. This device produces a binary output which is 1 if any of the devices in layer 3 of the column are active, otherwise 0. The layer four device is connected to various devices in the competition layer 5.

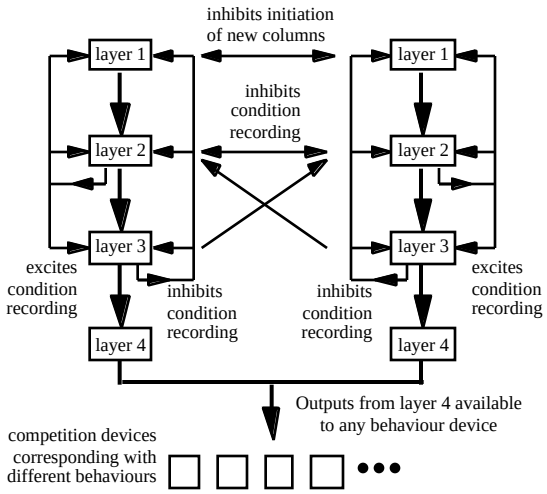

Fig. 2. Two columns within a clustering array, and outputs to competition. Specific layers within specific columns are the sources of inputs which excite and inhibit device condition recording in target columns.

Change management connectivity indicating the level of activity in the second layer excites change in every layer of the column and inhibits change in all other columns. Inter-column inhibition results in at most one column being selected as the location for change in response to one input state. Connectivity indicating the level of activity in the third layer inhibits change in all layers. Connectivity indicating the level of activity in the first layer inhibits initiation of new columns which have not yet commenced portfolio definition. The driving force for change is lack of output from clustering in response to an input state: a clustering array must always generate some output for interpretation as recommendations by competition.

Condition definition within an array proceeds as follows. System experience is divided into periods of exposure to sequences of input states ("wake" periods) and periods in which provisional condition defining connectivity is established ("sleep" periods). In the first wake period, no column responds in any way. In the first sleep period, groups of condition defining provisional inputs are generated by random selection, but within one column there is a statistical bias on the groups generated for devices in the first layer in favour of individual system inputs to which frequently occurred together in the same input states in the preceding wake period. In the subsequent wake period, the first input state containing a high proportion of these favoured inputs will generate condition recording until a layer 3 output is present.

Additional columns will be generated during future sleep periods if some input states in the preceding wake period generated no significant activity in the first layer of

any existing column. The new column will be configured with a statistical bias in its first layer in favour of system inputs which often occurred together in those input states, and will be initiated in the subsequent wake period if an generates no significant activity in the first layer of any existing column and contains a high proportion of the favoured system inputs. A newly configured column must have generated an output in response to a minimum number of input states before a new column can be configured. This delay reduces the risk of two columns being defined with very similar condition portfolios.

After initiation, if there is activity in layer 3 of any column no changes will occur, the array is already producing an output indicating that the input state corresponds with the column portfolio. If there is significant activity in layer 2 of some columns but no layer 3 activity, new conditions will be added to all layers of the column with the strongest layer 2 activity until a layer 3 output from that column is present. In other words, activity in layer 2 indicates that the input state is sufficiently similar to be added to the column portfolio.

One problem with these algorithms is that layer 2 activity could call for change in a column, but the available groups of provisional connections might not contain large enough active subsets to provide the additional conditions required. The probability of this situation occurring is reduced as follows. Groups of provisional connections are defined in every sleep period in each layer of every column. Connections are selected at random, but with a statistical bias in favour of inputs which have often been part of conditions recently detected in the same layer of the same column as the one within which the group is being defined. This bias also means that additional conditions will have a strong degree of similarity to existing conditions in the same module. Another problem is that a column might very rarely or never respond to an input state after the first. During sleep, the thresholds of first and/or second layer devices in such a column are reduced, until the column is generating outputs in response to a significant proportion of input states.

Although one possible implementation of the sleep mechanism involves a partial rerun of past experience [4; 5], the role of the mechanism is radically different from the rehearsal of past learning as a way to reduce disruption of past memories by new learning [20]. The recommendation architecture sleep mechanism configures appropriate resources for future learning on the basis of past experience but does not directly affect past learning. Sleep minimizes information exchange between modules and therefore reduces the side effects of later learning on prior learning [5].

A large input space is thus compressed into a smaller set of column condition portfolios. The conditions detected by one column are similar in information terms, and the presence of a condition within the portfolio is indicated by an output from the column. New columns are added and column portfolios expanded until every input state contains conditions in one or more portfolios. Because column portfolios are conditions at a higher level of complexity than inputs, they tend to discriminate better between different categories of input states. Portfolios can acquire many different behavioural meanings, but because column portfolios can expand by addition of similar conditions but cannot contract, the effects of portfolio changes on existing behavioural meanings are limited.

Competition devices have excitatory inputs from the devices in clustering layer 4. These inputs have continuously variable weights and devices produce outputs proportional to the sum of the inputs weight of all active inputs. The behaviour corresponding with the competition device with the largest output is the selected behaviour.

Weights are adjusted by feedback of two types. In supervised feedback, the component corresponding with the correct behaviour is indicated. A connection is established from any active layer 4 devices to that component if no connection already exists, and giver a standard weight. If the total of the weights of all active inputs to the device corresponding with the target behaviour is less than the total for any other device, those weights are all increased by the same proportion until the output from the target device is the largest.

Consequence feedback simply indicates whether the selected behaviour was correct or incorrect. If correct, the weights of active inputs to the competition device with the largest output are increased by 5% of the standard weight. If incorrect, the weights of those active inputs are decreased by 10% of the standard weight.

The competitive layer thus associates different combinations of column portfolios defined by clustering with different behaviours. If these columns do not provide adequate discrimination between input states appropriate for different behaviours, as indicated by lack of convergence on consistently positive consequences, a number of approaches can be followed. One is to use information on the "degree" to which cluster portfolios are present, as indicated by the number of third level devices in the column that are active. The second is to use the identities of these third level devices to discriminate between different conditions within the column portfolio, in other words to give each third level device a separate weight into the fifth level. The third approach is to force the column to use additional third level devices to record conditions when the column is already producing an output. These additional devices will provide additional discrimination, which could operate through one additional fourth level device which only produced output in response to activity of new third level devices. A fourth approach is to add columns in a similar input space to that covered by columns which are producing outputs with inadequate discrimination.

All these approaches use available information that the outputs of some column or group of columns are not converging on consistently positive consequences to expand the set of conditions detected in the input space of the appropriate columns, but retaining the capability to always produce an output in response to an exact repetition of an input state which generated an output in the past.

V. Test Scenario and Results

In order to investigate the effects of later learning on prior learning, a test scenario was devised. The concept in this scenario was that a learning system was presented with a series of conditions (or objects). The presence or absence of 1000 characteristics could be discriminated in each object, and the information available to the system from each object was therefore equivalent to a 1000 bit binary vector. Such an information content is roughly equivalent to 100 input variables accurate to three significant figures. Input objects were artificially constructed. Objects (i.e. input states) were defined by the set of characteristics which they possessed, and an object was represented as a one thousand element binary vector with an element being one or zero depending on the presence or absence of the corresponding characteristic in the object. Sixty different categories of input object were defined by probability distributions for the occurrence of characteristics in instances of the category. Individual instances of object categories were created by random selection from a population of characteristics with the appropriate distribution. The same object was never presented to the system more than once.

The system first learned to recognize 50 categories, then was challenged with an additional 10 categories. Accuracy in response to the initial 50 categories was ~90%. After learning of the extra ten categories, the recognition of the new categories was ~90% and the recognition of the old categories was reduced to ~80%. This degradation was gradual and also graceful in the sense that in all cases of incorrect recognition the second choice of the system was the correct category.

VI. Conclusions

The stability-plasticity dilemma limits the ability of machine learning to solve operationally complex problems. The maintenance of adequate meanings for device outputs as learning proceeds is at the root of the stability-plasticity dilemma, and that the type of algorithms used in conventional machine learning approaches make it difficult to maintain such meanings.

An alternative type of device algorithm which allows maintenance of a higher degree of meaning during learning and the recommendation architectural framework which is able to use this type of algorithm have been described. The differences between this framework and other approaches have been brought out, and the conclusion reached that the recommendation architecture approach can retain higher degrees of prior operational meaning during learning. Systems using this architectural approach therefore have greater potential for solving operationally complex problems.

References

[1] Anderson, C.W. and Hong, Z. (1994). Reinforcement Learning with Modular Neural Networks for Control. *Proceedings of the IEEE International Workshop on Neural Networks Applied to Control and Image Processing*.

[2] Anderson, J.A. and Sutton, J.P. (1997). If we compute faster, do we understand better? *Behavior Research Methods, Instruments & Computers* 29 (1), 67-77.

[3] Carpenter, G.A. and Grossberg, S. (1988). The ART of Adaptive Pattern Recognition by a Self-Organizing Neural Network, *IEEE Computer*, 3, 77-88

[4] Coward, L. A. (1990). Pattern Thinking. Praeger: New York.

[5] Coward, L. A. (2000). A Functional Architecture Approach to Neural Systems. *Int. Journal of Systems Research and Information Systems*, 9, 69-120, 2000.

[6] Coward, L.A., Gedeon, T. and Kenworthy, W. (2001). Application of the Recommendation Architecture to Telecommunications Network Management. *Int. Journal of Neural Systems* 11(4) 323-327.

[7] Coward, L.A. (2001). The Recommendation Architecture: lessons from the design of large scale electronic systems for cognitive science. *Journal of Cognitive Systems Research* 2(2), 111-156

[8] Crutchfield, J.P. (1998). Dynamical Embodiments of Computation in Cognitive Processes. *Behavioural and Brain Science* 21, 635-637.

[9] French, R.M. (1999). Catastrophic Forgetting in Connectionist Networks. *Trends in Cognitive Science* 3(4), 128-135.

[10] Gedeon, T., Coward, L. A., and Zhang, B. (1999). Results of Simulations of a System with the Recommendation Architecture, *Proceedings of the 6th International Conference on Neural Information Processing*, Volume I, pages 78-84.

[11] Gokcay, E. and Principe, J.C. (2002). Information Theoretic Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24(2) 158-169

[12] Hebb, D.O. (1949). *The Organization of Behaviour*. New York: Wiley.

[13] Jordan, M.I. and Jacobs, R.A. (1994). Hierarchical Mixtures of Experts and the EM Algorithm. *Neural Computation* 6, 181 - 214.

[14] Klir, G.J. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. NJ: Prentice Hall.

[15] Kohonen, T. (1995). *Self Organizing Maps*. Springer Series in Information Sciences 30.

[16] Nortel Networks (2001). *DMS-100/500 Feature Planning Guide* http://www.nortelnetworks.com/products/01/dms100w/doclib.html

[17] Ludermir, T., Carvalho, A., and de Souto, M. (1999). Weightless Neural Models: A Review of Current and Past Works. *Neural Computing Surveys* 2, 41 - 61.

[18] Ratnayake, U. and Gedeon, T.D. (2002). Application of the Recommendation Architecture for Discovering Associative Similarities in Documents. *Proceedings of the 9th International Conference on Neural Information Processing*.

[19] Ratcliff, R. (1990). Connectionist Models of Recognition Memory: Constraints imposed by learning and forgetting functions. *Psychological Review* 97, 285 - 308.

[20] Robins, A. (1995). Catastrophic Forgetting, rehearsal, and pseudorehearsal. *Connection Science* 7, 123 - 146.