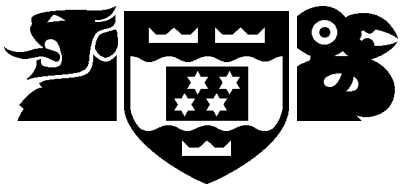
VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wananga o te Upoko o te Ika a Maui



Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

An ART-based Modular Architecture for Learning Hierarchical Clusterings

G. Bartfai

Technical Report CS-TR-95/3
February 1995

Abstract

This paper introduces a neural architecture (HART for "Hierarchical ART") that is capable of learning hierarchical clusterings of arbitrary input sequences. The network is built up of layers of Adaptive Resonance Theory (ART) network modules where each layer learns to cluster the *prototypes* developed at the layer directly below it. The notion of *effective vigilance* is introduced to refer to the vigilance level of multiple ART modules in a HART network. An upper bound is derived for the number of HART layers needed in the case when all ART modules have the same vigilance. Experiments were carried out on a machine learning benchmark database to demonstrate the developed internal representation as well as some learning properties of two- and three-layer binary HART networks.

Keywords: Adaptive Resonance Theory, Self-organization, Hierarchical clustering, Machine learning, Zoo database.

Publishing Information

This paper has been accepted for publication in *Neurocomputing*.

Author Information

Guszti Bartfai is with the Department of Computer Science, and his e-mail address is guszti@comp.vuw.ac.nz

1 Introduction

The ability to learn about the environment without a teacher has long been considered an important characteristic of intelligent systems. Unsupervised learning can be found both at sensory-level of mammals, and at higher, cognitive levels of humans. Therefore, it has been an important topic in neural network research (e.g. [13], [17], [18], [11]) in recent years. Unsupervised learning networks typically perform dimensionality reduction or pattern clustering. In the latter case, upon presentation of an input pattern, the node whose connection weight vector is the closest to the current input vector (in some distance metrics) will become the only active node (winner-take-all competition), and will be allowed to learn, i.e. modify its connection weights. After repeated exposure to the input environment, the network will store a prototypical element of each category, or class, in its connections weights. The number and size of classes the network finds on a given training set depends on the number of output nodes. The more neurons the network has, the larger number of more specific categories it will find. Because of its "winner-take-all" characteristics, the network will only be able to find *one* category for a given input pattern, regardless of how large the network is[1]. However, environments are often more complex and may exhibit a hierarchical structure (e.g. classification of animals), which humans can learn without difficulty. It is, therefore, worthwhile investigating neural network architectures that can learn class hierarchies. Networks that contain multiple competitive layers appear to be suitable candidates. For example, Rumelhart et al. [18] suggest that a hierarchical structure can be formed if a network has several "inhibitory clusters" in each layer that work in parallel, and each of them receives input from the output the previous layer only. Another typical example for networks with multiple competitive layers is the *cognitron* [10]. For the systematic construction of modular networks, however, Adaptive Resonance Theory (ART) neural network architectures [3] appear to be more suitable because of their well-defined interfaces as well as features that most other networks lack. In specific, ART networks have the ability to create new output nodes (i.e. categories) dynamically, and do not suffer from the problem of forgetting previously learned categories if the environment changes. They too, however, can only develop input categories at a given level of specificity, which depends on a global parameter called *vigilance*. There have been several attempts to combine ART modules in order to represent class hierarchies [9, 20, 19, 2].

In this paper, we describe a modular multi-layer network architecture built up of ART networks (*HART*, for "*Hierarchical ART*"), which is capable of developing hierarchical class representation through self-organisation. In a HART network, each layer — which is essentially an ART network — learns to cluster the *category prototypes* developed at the layer directly below it. This way, successively higher layers are able to gain more general "views" of the input environment while lower layers learn more specific categories. Due to its simple connectivity, the HART network retains the stability properties of ART networks. Some other useful properties of the architecture are also demonstrated both through analysis and experiments.

Section 2 briefly summarises the main features of ART neural networks that are sufficient for understanding the rest of the paper. The architecture and information processing of the HART network are described in section 3. The notion of *effective vigilance* is introduced in section 3.2, and an analysis reveals the upper bound on the number of layers in a HART network. Experiments that were carried out on a machine learning database are discussed in section 4, and conclusions are drawn in section 5.

[1] We note here that competitive networks perform, in general, *contrast enhancement* [3], which may help overcome some of the above limitations of "winner-take-all" networks.

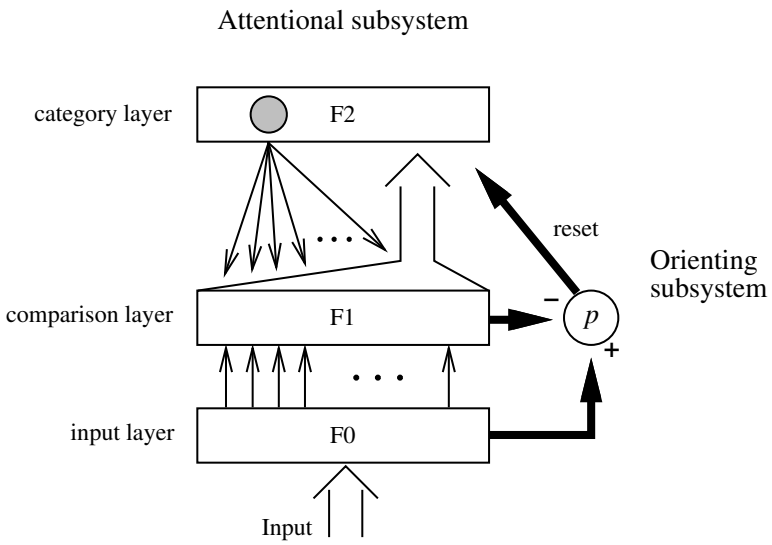


Figure 1: Architecture of the ART network. Thick arrows denote combined or non-specific signals. Winning F2 category nodes are selected by the *attentional* subsystem. Category search is controlled by the *orienting* subsystem. If the degree of category match at the F1 layer is lower than the vigilance level ρ, the non-specific reset signal will be triggered, which will deactivate the current winning F2 node for the period of presentation of the current input.

2 ART neural networks

This section describes the ART architecture at a level of detail that is necessary for understanding the rest of the paper. Unless otherwise stated, we refer to ART1, the binary version of ART [3][2].

Adaptive Resonance Theory (ART) architectures are neural networks that develop stable recognition codes by self-organisation in response to arbitrary sequences of input patterns. They were designed to solve the "stability-plasticity dilemma" that every intelligent machine learning system has to face: how to keep learning from new events without forgetting previously learned information.

An ART network is built up of three layers: the *input* layer (F0), the *comparison* layer (F1) and the *recognition* layer (F2) with N, N and M neurons, respectively (see figure 1). The input layer stores the input pattern, and each neuron in the input layer is connected to its corresponding node in the comparison layer via one-to-one, non-modifiable links. Nodes in the F2 layer represent input categories. The F1 and F2 layers interact with each other through weighted bottom-up and top-down connections that are modified when the network learns. There are additional gain control signals in the network (not shown in figure 1) that regulate its operation, but those will not be detailed here. The learning process of the network can be described as follows.

At each presentation of a non-zero binary input pattern $\mathbf{x}$ ($x_j \in \{0,1\}$, $j = 1,2,...,N$), the network attempts to classify it into one of its existing categories based on its similarity to the stored prototype of each category node. More precisely, for each node i in the F2 layer, the bottom-up activation T_i is calculated, which can be expressed as

[2]There are ART networks that accept both continuous and binary inputs [4, 7].

$$T_i = \frac{|\mathbf{w}_i \cap \mathbf{x}|}{\beta + |\mathbf{w}_i|} \quad i = 1, ..., M \tag{1}$$

where $|\cdot|$ is the norm operator ($|\mathbf{x}| \equiv \sum_{j=1}^{N} x_j$), $\mathbf{w}_i$ is the (binary) weight vector (or prototype) of category i [3], and $\beta > 0$ is the *choice parameter* [6]. Then the F2 node I that has the highest bottom-up activation, i.e. $T_I = \max\{T_i | i = 1, ..., M\}$, is selected (winner-take-all competition). The weight vector of the winning node ($\mathbf{w}_I$) will then be compared to the current input at the comparison layer. If they are similar enough, i.e. they satisfy the

$$\frac{|\mathbf{w}_I \cap \mathbf{x}|}{|\mathbf{x}|} \geq \rho \tag{2}$$

matching condition, where ρ is a system parameter called *vigilance* ($0 < \rho \leq 1$), F2 node I will capture the current input and the network learns by modifying $\mathbf{w}_I$:

$$\mathbf{w}_I^{new} = \eta(\mathbf{w}_I^{old} \cap \mathbf{x}) + (1 - \eta)\mathbf{w}_I^{old} \tag{3}$$

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

If, however, the stored prototype $\mathbf{w}_I$ does not match the input sufficiently, i.e. condition (2) is not met, the winning F2 node will be reset (by activating the reset signal in figure 1) for the period of presentation of the current input. Then another F2 node (or category) is selected with the highest T_i, whose prototype will be matched against the input, and so on. This "hypothesis-testing" cycle is repeated until the network either finds a stored category whose prototype matches the input well enough, or allocates a new F2 node. Then learning takes place according to (3).

It is important to note that once a category is found, the comparison layer (F1) holds $|\mathbf{w}_I \cap \mathbf{x}|$ until the current input is removed. It can be shown that after an initial period of self-stabilisation, the network will directly (i.e. without search) access the prototype of one of the categories it has found in a given training set.

Note also, however, that, as a consequence of its stability-plasticity property, the network is capable of learning "on-line", i.e. refining its learned categories in response to a stream of input patterns (as opposed to being trained "off-line" on a finite training set).

The number of developed categories can be controlled by setting ρ: the higher the vigilance level, the larger number of more specific categories will be created. (If $\rho = 1$, the network will create a new category for every unique input.)

In order to get two or more "views" of the training set with the same network, one can consider choosing low vigilance first to get an overall view, and then raise it at a later phase during training to gain a more detailed view as well. This, however, will result in the network forgetting the overall view and (i.e. coarser categorisation) retaining only the more specific categories, even if vigilance is lowered again later. This is because only one category is selected for a given input, and if two categories ($\mathbf{w}_l$, $\mathbf{w}_h$) were found to match the current input perfectly, i.e.

$$|\mathbf{w}_l \cap \mathbf{x}| = |\mathbf{w}_l|, \quad \text{and} \quad |\mathbf{w}_h \cap \mathbf{x}| = |\mathbf{w}_h| \ ,$$

the more specific one (i.e. of the larger norm) will be preferred due to (1). There is no relationship between any pairs of category prototypes except that they are in the same network and compete with each other. The network, therefore, with its single layer of category nodes, is not capable of representing (and thus learning) a hierarchy of classes.

[3] For simplicity, we assume that the network is in fast learning mode ($\eta = 1$, see below), in which case the bottom-up and top-down weights for each F2 category i are identical ($= \mathbf{w}_i$, see [3]).

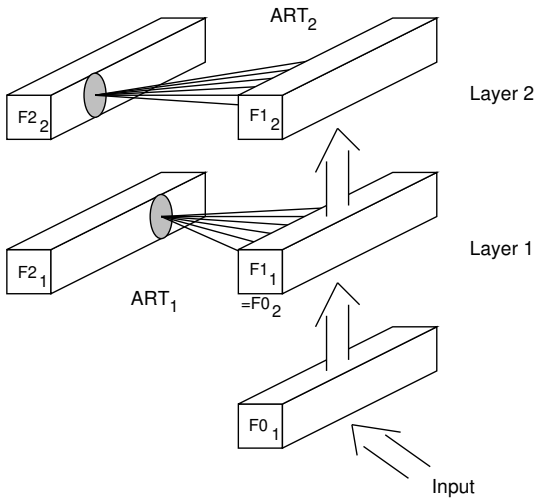


Figure 2: Architecture of a two-layer HART network. The F0 $\rightarrow$ F1 connections are unidirectional, one-to-one, and non-modifiable, as in ART. For clarity, only the connections from the F1 layer to the active (winning) F2 node in each HART layer are shown. This network is capable of developing a two-level class hierarchy from a sequence of input patterns.

3 The HART network

The HART network we propose here is a modular, multi-layer architecture that can be used to develop hierarchical clusterings of arbitrary sequences of input patterns. It is composed of L layers of ART network modules with ART_1 and ART_L being the bottom and top layers, respectively. The layers in HART are connected simply by uniting the F0 layer of the ART_l module and the F1 layer of the HART layer below (i.e. ART_{l-1}). In other words, each layer in a HART network receives its input only from the F1 (comparison) layer of the HART layer directly below it (via one-to-one non-modifiable links as discussed in section 2).

More precisely, the l-th layer in a HART network is an ART network (ART_l) with layers $\mathrm{F0}_l$, $\mathrm{F1}_l$ and $\mathrm{F2}_l$, and vigilance parameter ρ_l. The sizes of the $\mathrm{F0}_l$, $\mathrm{F1}_l$ and $\mathrm{F2}_l$ layers are N_l, N_l and M_l, respectively. According to the above,

$$F0_l \equiv F1_{l-1}, \quad l = 2, \ldots, L \tag{4}$$

where L is the number of layers in the HART network. Also,

$$N_l = const\ (= N), \quad l = 1, ..., L \tag{5}$$

due to (4).

The architecture of a two-layer HART network is shown in figure 2.

3.1 Training the network

The learning process of the HART network is described as follows.

Step 1: Initialisation

The network is initialised by resetting each ART module. No further initialisation is needed.

Step 2: New input
A new input vector ($\mathbf{x}$) is presented to the network by registering it into the $F0_1$ layer.

Step 3: Category search
The ART_1 module starts searching for an appropriate category in the $F2_1$ layer. Layer $F1_1$ contains $\mathbf{w}_I \cap \mathbf{x}$ once ART_1 has found node I in layer $F2_1$ to match the input pattern sufficiently according to (2).
The ART_2 network (i.e. layer 2 in HART) can then start searching for a suitable category for its input vector, which is now $\mathbf{w}_I \cap \mathbf{x}$.
This search process goes on involving successively higher layers until the ART module at the highest layer (ART_L) finds an appropriate category for its input.

(This sequential order of search can be implemented by appropriate *gain control* signals [3] to the $F1$ and $F2$ layers such that the orienting subsystem at layer l will be activated only when the adaptive search process has completed at layer $l-1$. Note, however, that this is not absolutely necessary for the correct operation of the network. The network will be able to operate correctly even if all HART layers are enabled throughout the search process. In this case, the input to the ART modules at higher levels will be changing rapidly as search takes place at lower levels, which prevents them from reaching a stable (i.e. resonant) state. Once the network at a lower level has resonated, only then will the next level be able to complete its search process. The above assumption of pure sequential search will not affect the outcome of the learning process, but can simplify computer simulations.)

Step 4: Learning
Weights of the winning category node in each HART layer are adjusted according to (3).

Steps 2 to 4 are repeated for each new input.

(Note that, in a HART network that is built up of general ART network as specified in [3], steps 3 and 4 can be interleaved: once ART_l has found a matching category, it can start adapting its weights while layer $l+1$ — and possibly higher layers, too — is in the middle of the search process. To achieve this, the ART network dynamics [3] can be adjusted by appropriate parameter settings.)

It can be shown that the learning properties of ART1, which were discussed in [15], apply also to the HART network as a direct consequence of the stability properties of the component ART1 networks. Most notably, after a finite number of presentations of a (binary) training set, each pattern will directly (i.e. without search) read out its corresponding categories in the HART network.

(We also note here that, since each layer learns from the prototypes the previous layer has developed, layer l can at most develop as many categories as layer $l-1$. Therefore, ART networks of the same capacity (or F2 layer size) can always be used in a HART network.)

As a result of the above learning mechanism, successively higher levels will develop fewer, and more general categories of the *input patterns*. Therefore, the output of the HART network, which is composed of the outputs of the $F2_l$ layers, will at any time show which class the current input belongs to *at each level in the class hierarchy*. Furthermore, the prototypes of these categories are available, too, which can be used, for example, to locate features that are *not* inherited from the more general class at a higher level.

It may also appear as if the vigilance parameters (ρ_l) had to be decreased at higher levels in order to have increasingly broader (or more general) categories in the hierarchy. The following section shows that this is not necessarily so.

3.2 The *effective vigilance* of HART networks

This section provides an analysis of the effect of vigilance levels on the categorisation properties of the HART network.

Let us assume the HART network has L layers. The input vector to layer l is denoted by $\mathbf{x}^l$. The input vector to the network is $\mathbf{x}$ ($\equiv \mathbf{x}^1$). Let $\mathbf{w}_i^l$ denote the weight vector, or category prototype, of the i-th F2 node in the l-th layer. Also the vigilance level of layer l is denoted by ρ_l.

After self-stabilisation, no further learning occurs, therefore

$$|\mathbf{w}_I^l| = |\mathbf{w}_I^l \cap \mathbf{x}^l| \quad l = 1, \ldots, L \tag{6}$$

necessarily, where $\mathbf{w}_I^l$ is the weight vector of the winning node I in layer l when $\mathbf{x}^l$ is presented.

From (6) and (2), it follows that in each layer

$$|\mathbf{w}_i^l| \geq \rho_l |\mathbf{x}^l|_{min} \quad i = 1, \ldots, M_l \quad l = 1, \ldots, L \tag{7}$$

where $|\mathbf{x}^l|_{min}$ is the smallest norm of all the inputs to layer l during learning on a given training set. However,

$$\mathbf{x}^l = \mathbf{w}_I^{l-1} \cap \mathbf{x}^{l-1} \tag{8}$$

due to the architectural constraint of the HART network as expressed in (4).

Substituting $\mathbf{x}^l$ in (7) with $\mathbf{x}^l$ in (8), and applying (7) to layer $l-1$ noting (6), we get

$$|\mathbf{w}_i^l| \geq \rho_l \rho_{l-1} |\mathbf{x}^{l-1}|_{min} \ . \tag{9}$$

Applying (9) recursively, the norm of the i-th F2$_l$ node $|\mathbf{w}_i^l|$ can be expressed as

$$|\mathbf{w}_i^l| \geq \prod_{j=1}^{l} \rho_j \cdot |\mathbf{x}|_{min} \quad i = 1, \ldots, M_l \quad l = 1, \ldots, L \ . \tag{10}$$

Let $\hat{\rho}_l$ be the *effective vigilance* of layer l, and be defined as

$$\hat{\rho}_l \equiv \prod_{j=1}^{l} \rho_j \ . \tag{11}$$

Note that $\hat{\rho}_l$ is the vigilance level of layer l with respect to the input of the *whole network* (i.e. $\mathbf{x}$, or $\mathbf{x}^1$).

Since $0 < \rho \leq 1$, it follows from (10) that the addition of a new layer can only decrease the effective vigilance of all the layers above it, including that of the new top layer ($\hat{\rho}_L$).

Note that $\hat{\rho}_L$ approaches zero even if ρ_l increases with l, or if all ρ_l are equal (and $\rho_l \neq 1$). This may look counterintuitive, but it is a direct consequence of the way the ART modules are interconnected: each layer learns the *prototypes* of the previous layer, which can only be smaller in size than their input patterns. Just how quickly $\hat{\rho}_L$ approaches zero will depend on the actual values of ρ_l. Since the minimum size of a prototype vector is 1, adding new layers to a HART network will not make any difference once $\hat{\rho}_L |\mathbf{x}|_{min}$ gets below 1. Therefore, given the minimum size of the input patterns in a training set, and the individual ρ_l, we shall have an upper bound on the number of layers needed.

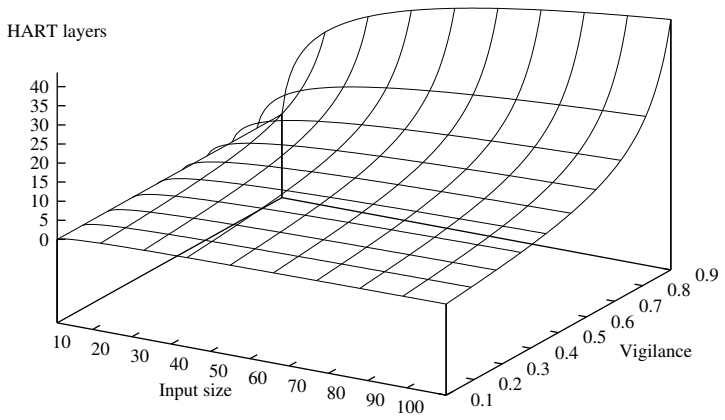


Figure 3: The maximum number of HART layers as a function of input size ($K \in [1, 100]$) and vigilance ($\rho \in [0.1, 0.9]$).

A simple analysis can be done if we assume all ρ_l are equal (i.e. $\rho_l = \rho,\ l = 1, ..., L$). In this case,

$$\hat{\rho}_l = \rho^l \ .$$

(Note the difference between ρ^l and ρ_l here.)

Furthermore, we assume all inputs are of the same norm, i.e. $|\mathbf{x}_k| = const = K$. This can be guaranteed by applying *complement coding* to the input patterns before presenting them to the network. Complement coding was introduced in [7] (and was also applied in [6]), and was shown to achieve desirable properties with ART networks, in particular to avoid the problem of *proliferation of categories* that was reported in [15].

From (10), the minimum size of a category prototype in layer l is

$$|\mathbf{w}^l|_{min} = \rho^l K \ \ , \tag{12}$$

and we look for the minimum integer n such that

$$\rho^n K < 1 \ \ . \tag{13}$$

In this case, having $L_{max} = \lfloor n + 1 \rfloor$ gives us the maximum number of layers needed. From (13), we get

$$n > -\frac{\log K}{\log \rho} \ \ . \tag{14}$$

For example, if $K = 20$ and $\rho = 0.5$ (i.e. $\hat{\rho}_l$ is halved at each successively higher layer), then $L_{max} = 5$. In other words, we do not need more than 5 layers to cover the broadest possible class that can be learned with a HART network.

Figure 3 plots the lower limit for n according to (14) in a range of input sizes (K) and vigilance levels (ρ). Since it would rarely be desirable to have 40 layers, we may well need to set the vigilance layers individually to gain more control over the granularity of the clustering. Alternatively, some adaptive technique could be used to adjust the vigilance levels, or even add new HART levels as necessary.

We also note that, since we looked at minimum prototype sizes only, the above analysis will hold even for situations when the network has not stabilised yet, i.e. not all inputs can read out their templates directly, which often occurs in real-world tasks.

4 Experimental results

This section presents experimental results that will demonstrate the main properties of HART network we have discussed above.

We carried out experiments on the "zoo" machine learning benchmark database [16]. It contains 101 instances of animals described with 18 attributes such as "hair", "aquatic", "domestic" and so on. The binary attributes were presented to the network in *complement coding* to assign significance to both the presence and absence of features, and to keep the input vectors normalised [6] to avoid proliferation of ART categories [15]. The only non-binary attribute ("number of legs") was presented in *generalised complement coding* [2]. The "type" attribute was ignored to eliminate any bias from the developed clusterings. The 18th attribute ("animal name") was simply used as a label for the individual instances. With this input encoding scheme, instances were presented to the network as 36-element binary vectors with the same norm of 16.

The simulations were carried out using a public domain neural network simulator program (`PlaNet` [14]) running under Unix and X-windows.

The network parameters were chosen to be similar to those presented in [6]. In particular, the initial values of the bottom-up weights in both ART modules were chosen in such a way that F2 nodes became active in the order $i = 1, 2, ...$ and were small enough so the network selected an uncommitted (or free) node only if $|\mathbf{w}_i \cap \mathbf{x}| = 0$ for all committed nodes. Also, the β choice parameters were sufficiently small that, among committed nodes, T_i was determined by the size of $|\mathbf{w}_i \cap \mathbf{x}|$ relative to $|\mathbf{w}_i|$. The network was also used in *fast learning* mode, i.e. during learning, the connection weights were allowed to reach their asymptotic values while the current input was presented.

In all the experiments, the HART network was trained "off-line" on a training set containing all 101 instances of the "zoo" database. In each training session, training proceeded by presenting the entire training set (in random order) repetitively (i.e. in *epochs*) until stability was achieved, i.e. inputs had direct access to their respective ART categories.

4.1 Internal representation of the HART network

In these experiments, we looked at the developed internal representation of the HART network.

Table 1 shows an example of the category attribute vectors that a two-layer HART network developed. It is one of the potentially many stable clusterings the network can discover in the training set. The actual clusterings vary depending on the order of presentation of the input patterns.

The hierarchical structure can be seen clearly: layer 1 categories 1-4 and 5-7 belong to layer 2 categories 1 and 2, respectively. The representative feature of super-class 1 is "venomous", which is inherited by all of its sub-classes. The same can be observed about attribute "milk" in super-class 2 and its sub-classes. Note also that although all sub-classes of super-class 1 share the same value for attribute "backbone", yet it is regarded as non-critical in that super-class (i.e. its attribute value is "don't care"). The network does not have any built in bias towards preferring "venomous" over "backbone" (or "yes" values over "no" for that matter). It simply means that at some stage during the training process, while the network was undergoing self-stabilisation, node 1 in layer 2 attracted an input with its "backbone" attribute "yes", which later ended up in super-class 2 (in one of sub-classes 6 or 7). Since the layer 2 vigilance level was low enough ($\rho_2 = 0.1$), the network accepted a generalised prototype for that class, keeping only "venomous" as critical. This situation cannot occur if

	Category prototype vectors								
Attribute	$\mathbf{w}_1^2$	$\mathbf{w}_1^1$	$\mathbf{w}_2^1$	$\mathbf{w}_3^1$	$\mathbf{w}_4^1$	$\mathbf{w}_2^2$	$\mathbf{w}_5^1$	$\mathbf{w}_6^1$	$\mathbf{w}_7^1$
hair	–	no	–	–	yes	–	yes	–	yes
feathers	–	yes	–	yes	–	–	yes	–	yes
eggs	–	yes	no	–	no	–	no	no	–
milk	–	no	–	–	yes	yes	yes	yes	yes
airborne	–	–	–	yes	yes	–	yes	–	yes
aquatic	–	–	–	–	–	–	no	yes	–
predator	–	–	–	–	–	–	–	–	no
toothed	–	no	yes	no	–	–	no	yes	–
backbone	–	no	no	no	no	–	no	–	–
breathes	–	no	no	–	–	–	–	no	–
venomous	yes	yes	yes	yes	yes	–	–	–	–
fins	–	–	yes	–	–	–	–	yes	yes
legs	–	–	–	–	–	–	–	–	–
tail	–	–	no	no	no	–	–	–	–
domestic	–	–	yes	yes	–	–	yes	–	yes
catsize	–	–	–	–	–	–	–	yes	–
category size	62	38	19	4	1	39	16	13	10

Table 1: An example for the developed (stable) internal representation of a two-layer HART network. A "–" attribute value should be interpreted as "don't care" (corresponding to a "00" pair in complement coding). Layer 1 and layer 2 vigilance levels were 0.4 and 0.1, respectively. In this training run, the network developed 2 and 7 categories at layers 2 and 1, respectively. The bottom row shows the number of input patterns that each category attracted. Note that the HART network does not have explicit connections between classes and their subclasses. The hierarchy "emerges" as inputs are presented and the corresponding category nodes at each layer are activated.

the network is trained in *sequential* mode (see section 4.2).

As another example, figure 4 depicts the class hierarchy that a three-layer HART network developed in a training session. It shows graphically how categories become increasingly specific (with more critical features) at lower levels. Note the empty boxes (3 and 4) in layer 2, which means that those categories were found identical to their parents (2 and 3 in layer 3) at the given vigilance level in layer 2. Only at layer 1 (with higher vigilance) were those categories split up into smaller, more specific ones. Some of the classes may seem rather odd (e.g. 5 in layer 1), by looking at three of the inputs they attract, and only "make sense" to us by identifying their distinctive features. This indicates that using the given 16 attributes, and without a bias or weightings of any of the features (which humans certainly have), those were *plausible* classes for the network to discover.

4.2 Learning properties of the HART network

We also carried out experiments, in which certain characteristics of the network as well as the training process were measured, and statistical quantities were calculated from a hundred different training runs.

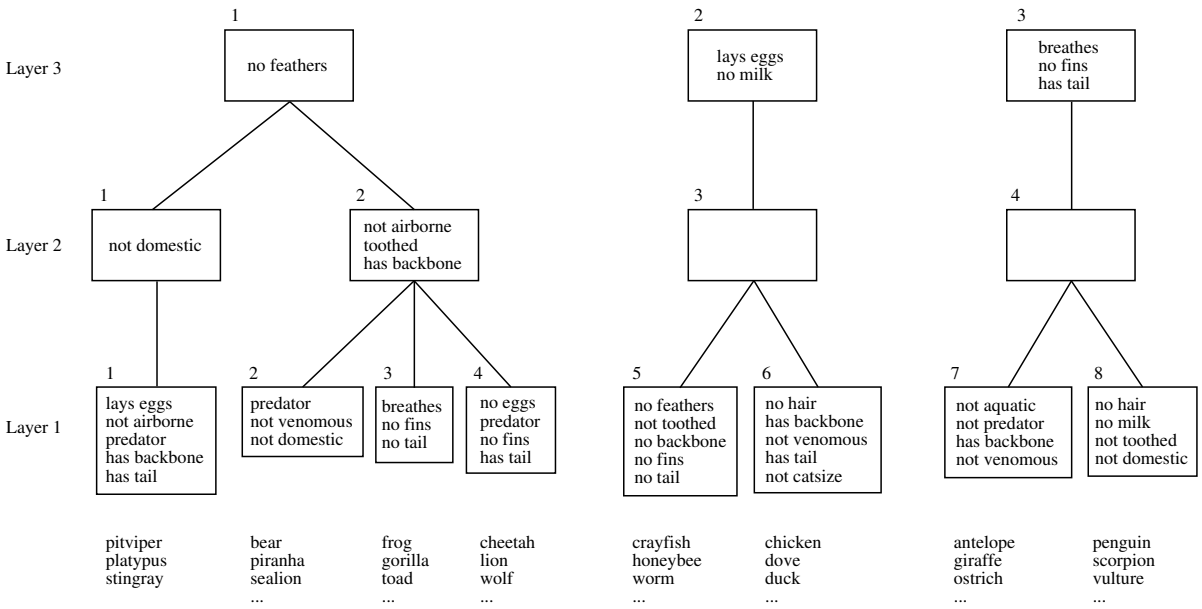


Figure 4: An example of a class hierarchy found by a three-layer HART network. Each box represents a category, and contains a list of critical and distinctive features of that class. All other attributes not shown were "don't care". For example, class 1 in layer 2 has "feathers" and "domestic" as critical features (both are absent) with "domestic" being the distinctive one (from class 2) at that level. Vigilance levels at layers 1, 2 and 3 were 0.4, 0.2 and 0.1, respectively. Below each layer 1 class are the names of three animals from the database that belong to that class.

Layers	Vigilance $\rho_{1,2}$	No. of categories min-max	No. of categories average	No. of epochs min-max	No. of epochs average
Layer 1	0.4	6-10	8.00	—	—
Layer 2	0.4	4-8	5.88	3-4	3.01
	0.2	3-6	4.05	2-3	2.98
	0.1	2-4	3.01	3-3	3.00

Table 2: The effect of changing layer 2 vigilance, while keeping ρ_1 fixed. Measurements were taken from 100 training sessions. It shows the number of categories each layer developed, and the number epochs it took for the network to stabilise.

Layer 2 vigilance	Average no. of categories parallel	Average no. of categories sequential	Average no. of epochs parallel	Average no. of epochs sequential
0.4	5.88	5.18	3.01	5.43
0.2	4.05	3.42	2.98	5.21
0.1	3.01	2.56	3.00	5.22

Table 3: The effect of learning mode (parallel vs. sequential) on the number of epochs and developed categories. Layer 1 vigilance (ρ_1) was kept at 0.4 and is not shown here since, being at the lowest level, it was not affected by these experiments. Measurements were taken from the same 100 training sessions as in table 2.

First, the vigilance level at layer 2 (ρ_2) was changed while ρ_1 was fixed. The results are shown in table 2.

Here the effect of lowering ρ_2 on the number of layer 2 categories can be seen. Note that the average number of categories drops from 8.00 to 5.88 even with ρ_1 and ρ_2 being equal. This is the effect of layer 2 learning from the *prototypes* of layer 1, which have their size reduced already (see the analysis in section 3.2). Also note the few number of epochs (around 3 on average) it took to learn the training set, and that it was independent of the layer 2 vigilance.

We also looked at the effect of two different training modes: *parallel* and *sequential*. In parallel mode, each layer in the network learns in every presentation of a new input. In sequential mode, each layer starts learning only when the previous layer has stabilised, i.e. stopped learning on the training set. Initial experiments indicated that a more compact representation could be developed this way by filtering out initial fluctuations of categories due to the network undergoing self-stabilisation. Sequential mode can also be envisioned "biologically plausible" by considering the different time scales of categorisation and learning, which effectively allows learning to occur only after categorisation has been achieved at all levels.

The results of these experiments with a two-layer HART network can be seen in table 3. It can be seen from the table, that the network developed, on average, about 15% less categories in sequential mode. However, the number of epochs needed to train the network increased by about 76%! Moreover, we can expect a linear increase in the number of epochs as more layers are added to the HART network.

From these experiments, we can conclude that although parallel learning does result in an increase of the categories, it is not significant compared to the increment in training time

(expressed in number of epochs).

5 Conclusion

In this paper, we introduced a hierarchical network architecture built up of ART modules that is capable of learning stable hierarchical clusterings of arbitrary sequences of input patterns. We have shown that a multi-layer HART network develops categories from the input sequence at successively increasing levels of generality. These were confirmed by the results of experiments with two- and three-layer HART networks when trained on a machine learning benchmark database. The internal representation of the network demonstrated a (2/3-layer) hierarchy of classes, where lower-level classes "inherited" features from ones at higher-levels, and had their own distinctive features as well. Comparative experiments with two different learning modes (parallel and sequential) were also carried out, and the parallel learning mode was found to be better overall than the sequential one.

We believe that the ideas and results presented here show the benefits of modularity in learning more complex relationships from the input environment. In particular, the HART network can be used in conjunction with other methods (e.g. [1, 5, 8, 12]) to design larger ART networks capable of complex pattern classification tasks.

We also suggest there are several ways the HART network can be extended. In the future, we are planning to investigate the following:

- Instead of using binary ART, the architecture can be built up of either ART2 [4] or Fuzzy ART [7] modules so the HART network can accept both binary and continuous inputs. The architecture defined in this paper does allow this extension.

- The developed clusterings can be made more robust, i.e. less dependent on the order in which inputs are presented, by using (some of) the component ART modules in slow learning mode ($\eta < 1$ in (3)).

We hope that more experiments with the HART network and its extensions on a range of learning tasks will enable us to understand its behaviour better, and will ultimately extend the repertoire of neural network models available for solving real-world problems.

Acknowledgements

I would like to acknowledge the work of both reviewers, and thank the very constructive comments of one of them in particular.

References

[1] Y. Asfour, G.A. Carpenter, S. Grossberg, and G. Lesher. Fusion ARTMAP: A neural network architecture for multi-channel data fusion and classification. In *Proceedings of the World Congress on Neural Networks*, volume 2, pages 210–215, 1993.

[2] G. Bartfai. Hierarchical Clustering with ART Neural Networks. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 2, pages 940–944. IEEE Press, 1994.

[3] G.A. Carpenter and S. Grossberg. A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing*, 37:54–115, 1987.

[4] G.A. Carpenter and S. Grossberg. ART2: Self-Organizing of Stable Category Recognition Codes for Analog Input Patterns. *Applied Optics*, 26(23):4919–4930, December 1987.

[5] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, and D.B. Rosen. Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, 3(5):698–713, September 1992.

[6] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. ARTMAP: Supervised Real-Time Learning and Classification of Nonstationary Data by a Self-Organizing Neural Network. *Neural Networks*, 4:565–588, 1991.

[7] G.A. Carpenter, S. Grossberg, and D.B. Rosen. Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, 4:759–771, 1991.

[8] G.A. Carpenter and W. Ross. ART-EMAP: A neural network architecture for learning and prediction by evidence accumulation. Technical Report CNS/CAS-TR-93-015, Boston University, Boston, MA, 1993.

[9] Thomas P. Caudell, Scott D. G. Smith, G. Craig Johnson, Donald C. Wunsch, and Richard Escobedo. An industrial application of neural networks to reusable design. In Thomas P. Caudell, editor, *Adaptive Neural Systems, The 1990 IR&D Technial Report*, pages 185–190. The Boeing Company, July 1991.

[10] K. Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3/4):121–136, November 1975.

[11] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26(23):4979–4984, December 1987.

[12] K. Iizuka. Neural pattern recognition on multichannel input representations. In *Proceedings of the World Congress on Neural Networks*, volume 4, pages 139–143, 1993.

[13] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[14] Yoshiro Miyata. *PlaNet, A Tool for Constructing, Running, and Looking into a PDP Network*, 1991.

[15] B. Moore. ART1 and Pattern Clustering. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 174–185. Morgan Kaufmann, San Mateo, CA, 1989.

[16] P. M. Murphy and D. W. Aha. UCI Repository of Machine Learning Databases [Machine-readable data repository]. Technical report, Department of Information and Computer Science, University of California, Irvine, CA, 1992.

[17] E. Oja. Neural networks, principal components, and subspaces. *International Journal of Neural Systems*, 1:61–68, 1989.

[18] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group. *Parallel Distributed Processing; Explorations in the Microstructure of Cognition*, volume 1: Foundations, chapter 5, pages 151–193. The MIT Press, 1986.

[19] Hui-Shin Vivien Soon and Ah-Hwee Tan. A Memory Model for Concept Hierarchy Representation and Commonsense Reasoning. Technical Report CAS/CNS-93-032, Boston University, Center for Adaptive Systems and Department of Cognitive and Neural Systems, January 1993.

[20] Ah-Hwee Tan. Adaptive Resonance Associative Map: A hierarchical ART system for fast stable associative learning. In *Proceedings of IJCNN*, volume 1, pages 860–865, June 1992.