Tina Memo No. 1997-002
and published in Neural Networks Journal, 10, 2, pp.315-326, 1997.

Supervised Learning Extensions to the CLAM Network.

N. A. Thacker, I.A.Abraham and P.Courtney.

Last updated
1 / 12 / 1998

This document forms part of the **Recognition and Intelligence** Series available from www.tina-vision.net.

2007-001	Retinal Sampling, Feature Detection and Saccades: A Statistical Perspective.
2006-008	Statistical Principles for Selection of Computer Vision Algorithms as Modules for Visual Perception - Show Me the Errors.
1991-001	Designing a Layered Network for Context Sensitive Pattern Classification.
1997-002	Supervised Learning Extensions to the CLAM Network.
1996-003	Tutorial: Algorithms For 2-Dimensional Object Recognition.
1997-005	Speechreading Using Probabilistic Models.
2000-002	Solving Shape Based Object Recognition from a Computational Standpoint - Practical and Physiological Constraints.
1995-004	Assessing the Completeness Properties of Pairwise Geometric Histograms.
1996-004	Robust Recognition of Scaled Shapes Using Pairwise Geometric Histograms.
1996-005	Multiple Shape Recognition Using Pairwise Geometric Histogram Based Algorithms.
2007-007	Automatic Identification of Morphometric Landmarks in Digital Images.
1999-002	A Feature Representation for Map Building and Path Planning.
2001-015	Colour Image Segmentation by Non-Parametric Density Estimation in Colour Space.
2001-006	What is Intelligence?: Generalised Serial Problem Solving.
1994-002	A Correlation Chip for Stereo Vision.
1995-001	Specification and Design of a General Purpose Image Processing Chip.

Imaging Science and Biomedical Engineering Division,
Medical School, University of Manchester,
Stopford Building, Oxford Road,
Manchester, M13 9PT.

1 Abstract

The Contextual Layered Associative Memory (CLAM) has been developed as a self-generating structure which implements a probabilistic encoding scheme. The training algorithms are geared towards the unsupervised generation of a layerable associative mapping [17]. We show here that the resulting structure will support layers which can be trained to produce outputs that approximate conditional probabilities of classification. Unsupervised and supervised learning algorithms operate independently permitting the unsupervised representational layer to be developed before supervision is available. The system thus supports learning which is inherently more flexible than conventional node labeling schemes.

Symbols used

$\sum$	summation
$S^{1/2}$	square root
$\int_{-\infty}^{\infty}$	integration between limits
$P(x\|w)$	probability of x given w
Δz	change in z
δ	delta
$f(S)$	function of S
I, J, K, L	unsupervised layers
i, j, k, l, m, n	nodes in an unsupervised layer
I_i	input into node i
o_i	output from node i
X	supervised layer
x	node in supervised layer
O_x	output from node x

2 Introduction

The field of neural networks constitutes a body of algorithms and structures designed to operate as learning systems. The specific learning regimes in which these systems are used can be divided into two distinct categories: unsupervised and supervised training. There seems to be a lack of proposals for network architectures which could be applied in both domains, despite the advantages that such a structure may offer of learning in situations when incomplete or limited external information is available. Training of the architectures may involve various amounts of prior knowledge about the problem domain or the required performance of the trained system. This prior knowledge is often hidden in the form of training parameters of the network system, for example learning rates, optimal number of nodes or architecture. These parameters can generally only be determined by trial and error, which can limit the application of such systems in general learning environments. One approach to this problem is to develop learning systems which can adapt themselves to the demands of particular problems via a process of continuous learning.

The CLAM network has been designed as a self-generating classification architecture which grows (generates nodes) to describe a particular input representation based on the expected (or observed) fluctuations in the input data. It is our belief that this information is likely to be more readily available than knowledge such as the optimal number of nodes required for a given network architecture applied to a particular problem. The network thus generates a representation of the input space with a granularity and thus accuracy driven by the expected accuracy of the data.

The network has been designed using the principle of information preservation. We feel this is desirable in order to enable hierarchical recognition, that is, the classification of sets of patterns that can themselves be classified as patterns. Multi-layering facilitates this in that sets of patterns can be the output of one layer, while sets of sets can be the output of subsequent layers. By information preservation here we simply mean the preservation of conditional probabilities in preference to simple winning-node output, as opposed to a formal information theoretic

meaning. In a previous paper [17] it was shown how probabilistic encoding of outputs (estimates with confidences) allows the layering of associative memories for such hierarchical recognition tasks.

The CLAM architecture is composed of two sorts of layers - unsupervised representational layers, which feed into supervised classification layers. The output response of nodes in the unsupervised representational layers are defined according to the probability that each could have been chosen as a maximum on the basis of the dot-product metric (reasons for the choice will be given later). This probability is proportional to the firing rate that would be obtained in a neuronal system operating a winner-take-all system such as that suggested by Grossberg in [4]. Furthermore, each associative layer is structurally identical to all others[1].

Here we show how such unsupervised representational layers can be used to provide input to a different sort of layer for classification purposes. These classification layers are used in conjunction with a supervised training regime which supports the calculation of conditional probabilities of classification. This contrasts with conventional node labeling methods generally adopted for associative classification which preserve less more information.

In developing the system it was considered important that the performance should be invariant, wherever possible, to principled changes in the input data. By this we mean that the assumptions made and constraints applied to the input data should be self-consistent and maintained throughout the system. For instance arbitrary partitioning of the input data should not alter network performance. An example of this in CLAM is the handling of handling of information made available in the form of frequency histograms. Such histograms consists of a number of bins defined with upper and lower boundaries. The contents of each bin represents the number of times an event has yielded a value between these boundaries. The size and number of bins are parameters of the data collection system but the actual location of their boundaries is arbitrary. We would regard this displacement of the defined positions of bin boundaries as a 'principled' change in the input data, which would alter individual bin contents but should not change network performance.

The information capacity of a fixed architecture will always be limited and we are interested here in developing an architecture which is capable of continual learning, thus a flexible architecture is required. Such flexible architectures would require algorithms permitting the addition and removal of nodes and connections thus altering the information capacity to match the task [2]. In particular we would want the network performance to be *stable* with respect to node addition and removal, in the sense that the classification ability of the network should remain broadly unchanged rather than degrade abruptly when a fresh node is added. Of course classification ability should improve as the weights of the new node are adjusted. Sparse connectivity achieves the goal of stability because it ensures local representation and thus limits the impact of changes to the few connected nodes. As a side effect such a network would train in a time which is of linear order with respect to the number of input data patterns since there are far fewer weights to adjust. This is an extremely desirable characteristic absent from many network architecture and algorithms.

One can summarise the desirable characteristics of a general learning network as follows:

- Independent supervised and unsupervised learning - to make use of limited amounts of prior knowledge as and when available
- Online training and continuous learning - prior knowledge to be made explicit and not hidden in training or architectural parameters
- Flexible architecture with automatic self-generation - to deal with the varying information capacity of different tasks
- Probabilistic outputs - to permit the multi-layering of networks required for hierarchical classification tasks
- Linear order learning time

The CLAM algorithms deliver a network architecture with these properties. This paper contains a summary of the essential CLAM algorithms and the extensions necessary to support supervised training - that is an unsupervised training and a node-generation algorithm; and a supervised training algorithm. The overall system is demonstrated by the classification of significantly overlapping data distributions. We show that in this task, (as with other supervised network architectures) the discrimination power of the network approaches optimality in the sense that the decision boundaries are positioned in accordance with the underlying probability distributions.

The previous paper [17] concentrated on the unsupervised aspects of the original CLAM which included temporal integration to learn sets of patterns. The present paper discusses the combined unsupervised and supervised architecture for learning conditional probabilities.

[1] a strategy which may be a sensible one in a physiological neural system for developmental reasons.
[2] although only node-generation is dealt with in CLAM at present.

The rest of the paper is organised as follows: section 2 recaps on the previous paper by describing the structure and function of the unsupervised layers for spatial and temporal integration in a form which gives resistance to node addition and removal, while section 3 explains the training algorithms for unsupervised learning and node-generation. Section 4 discusses the architecture of the supervised layer and the overall model of CLAM is presented in the form of a diagram. Section 5 outlines a simple test used to demonstrate the performance of the extended network; and section 6 closes the paper with a reminder of the main elements and gives pointers to the application of CLAM in the area of machine vision. Some remarks which are felt to relate this statistically-based architecture to physiological neuronal systems are made during the course of the paper in the form of footnotes.

3 Designing a Layered Associative Memory

The CLAM system is an extension of conventional associative network pattern classifiers allowing a layered classification architecture to be developed. Figure 1 shows an example of CLAM being used to recognise sets of patterns presented serially.

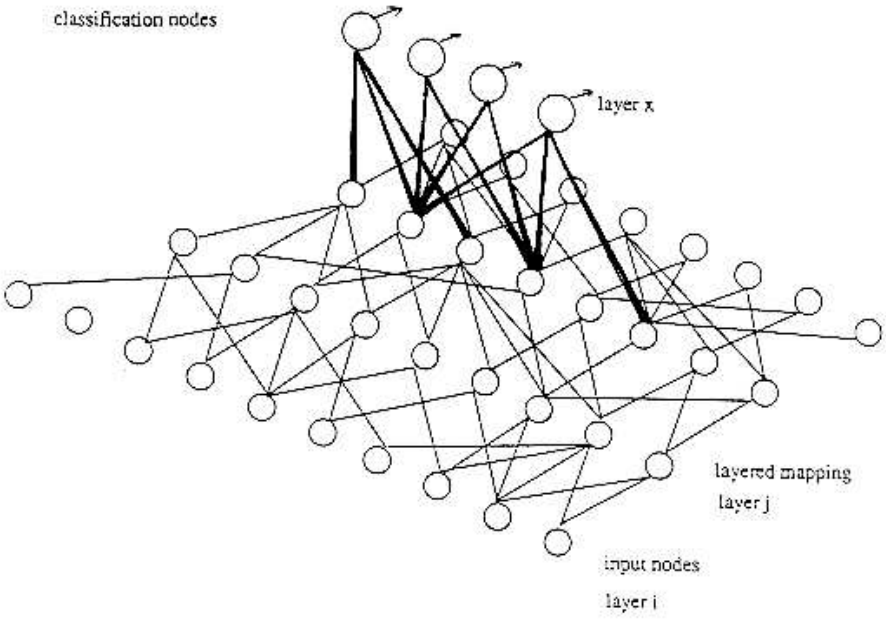

Figure 1: Network Architecture

The network algorithms are based on a simple stochastic model of neuronal function where the output of particular nodes is determined by the time-averaged response of a nearest-neighbour classification network to noisy input data and noisy internode connections (weights). In what follows we assume that the effects of such noise can be represented with mean values of input I_i and connection strength z_{ij} and their respective uncertainties δI_i and δz_{ij}. The classification scheme requires computation of the probability P_j that the input pattern would have best matched the pattern encoded at each node j on the basis of a Bayesian norm (or dot-product) of the I_i and z_{ij} in equation 6.1. This is as used in Baysian classifiers and is one of the simplest possible functions of neurons and synapses used in neural networks.

$$D_j = \sum_i I_i z_{ij} \tag{1}$$

subject to the expected errors δI_i and δz_{ij}. This form was chosen in preference to a Euclidean distance measure because connections do not need to be maintained for components of the input pattern which are expected to be zero. It is therefore particularly suitable for a system which is to have a flexible architecture as it permits smooth variation in network performance subject to the addition and removal of nodes and connections by incremental training from zero. The standard Euclidean metric is not adequate as it is a distance measure that can only be defined in a space of fixed dimensionality and thus a fixed number of connected nodes. The weighted Euclidean metric, which can give smooth performance under variable dimensionality requires an additional mechanism to deal with the weights, whereas with the dot-product metric this is all handled automatically[3]. This dot-product must

[3]the dot-product also has the advantage over the Euclidean metric on the basis of physiological simplicity.

also generate normalised templates of input patterns on the inputs connecting to each node so that equation 6.2 holds.

$$\sum_i z_{ij}^2 = 1 \tag{2}$$

All networks use an implicit subjective function for information retrieval [9] from which constraints on the properties of suitable input patterns can be derived. In the case of CLAM, the use of the dot-product implies that the input vector must have components which are a measure of the relative significance of each feature to the representation. We suggest that the logical extension to this line of reasoning is that *equal amounts of evidence* in different parts of the input space should carry *equal influence* in the selection of a maximum node. Referring back to the example of data presented in the form of a frequency histogram, this is equivalent to saying that the initial definition of input space partitions due to the positioning of the histogram bin boundaries should not have an effect on the performance of the trained system. We conclude [Appendix 1] that if we are deriving inputs for the network from measures of relative significance S_i (defined as proportional to the probability of observing feature i) which add linearly, then the correct function of significance suitable for input to a network using the dot-product metric is the square root, equation 6.3.

$$I_i = S_i^{1/2} \tag{3}$$

It is important to note that this result has been derived purely on the basis of using the dot-product to index into stored patterns in a principled way, that is, applying the implicit constraints to restrict the form of the input. In this case propagating the constraint restricts the form of input function. Similar reasoning can be proposed with respect to the normalisation of z_{ij}, the stored weights in equation 6.2, enforced during training for the storage of all patterns with equal total significance.

The node activation function now becomes compared to the dot-product definition and replacing the stored pattern z_{ij} with a template of previous data $S_{ij}^{1/2}$ gives equation 6.4.

$$D_j = \sum_i S_i^{1/2} S_{ij}^{1/2} \tag{4}$$

This discrete measure is a form of the Bhattacharyya distance measure L_B [8] for comparing continuous probability density functions $P(x|w_j)$ when S_i and S_{ij} equate to the two probability density functions in equation 6.5.

$$L_B = -ln \int_{-\infty}^{\infty} (P(x|w_1)P(x|w_2))^{1/2} dx \tag{5}$$

This is important as any variable we may wish to use for input to CLAM may be represented in the form of a probability density distribution, for example statistical frequency occurrence histograms of a measurement variable[4]. This process can be considered as analogous to the process of fuzzification often described in the fuzzy logic literature, but here as we will show the method can be justified in the context of a strictly Bayesian formalism.

As has been demonstrated elsewhere [6] choosing the node with the largest D_j is equivalent to choosing that with the lowest χ^2 chi-squared statistic - the common maximum likelihood estimator.

If we consider a simple neuronal model employing a winner-take all mechanism, the frequency output of each node will be constant for constant input frequencies and must be proportional to the node output probabilities P_j. The probabilistic form of the output allows a greater amount of information regarding the input pattern to be preserved. It has the rather unusual feature when compared to most other network models that the output of any one node is not simply a function of its inputs. This is due to the winner-take-all strategy used in the underlying stochastic model to generate the time-averaged output probabilities. These probabilities P_j can be generated in all cases by Monte-Carlo simulation of the noise fluctuations, but can be approximated analytically (though using an empirical formulation) assuming that all fluctuations in frequency-coding within the network are uncorrelated and normally distributed [Appendix 2].

The probablistic interpretation is not a complete specification of the output o_j as it must clearly also be dependent upon the magnitude of the input. The CLAM network has been designed to be layerable so that the time integrated outputs (mean output firing frequencies) from one layer can be used as inputs to the next. The buffered output from an associative layer thus has the correct statistical properties to be used in this way. Layered associative memories make possible hierarchical classification for representing and recognising sets of input vectors. When layering the network, the output is integrated over time and must produce values which are invariant to temporal segmentation of the input. That is, the total number of pulses propagated through each part of the network must

[4]This is may be likened to the physiological phenomenon of electrical pulses along an axon as a way of frequency-coding data. Each bin of the histogram can be thought of as an input to a neuron at a synapse, with the number of entries in a bin as the total amount of neuro-transmitter released by the neuron in a fixed time interval.

be the same regardless of the order of presentation of the patterns[5]. These constraints specify the analytic form of the output from each node o_j [Appendix 3] which is a measure of significance of the match between the input pattern and the stored weights in equation 6.6.

$$o_j = P_j |I|^2 \tag{6}$$

In the case of the winning node, the stored template and input pattern will be very similar thus the modulus of the input $|I|$ can be replaced with the locally computed D_j. Probabilistic encoding correctly normalises the output data so that correct account is taken of all components forming the input to the next layer. Input to a node k in the next layer is obtained in accordance with our previous result [Appendix 1] and is simply equation 6.7.

$$I_k = (\sum_t o_j)^{1/2} \tag{7}$$

where the summation implies a temporal addition of the outputs generated by all of the current set of patterns presented to layer J for classification by layer K. Thus the generated output is independent of the order of presentation of data to the previous layer.

4 Training Associative Layers

The associative layers of the network can be used as an interpolative memory system, using simple weighted averages of values V_j associated with each output node to predict the expectation value of the variable $< V >$. We have found that such an interpolation mechanism can be an order of magnitude more efficient at encoding output mappings than conventional nearest-neighbour systems. By this we mean that the interpolation can give better resolution for fewer nodes.

If we choose as in equation 6.8

$$V_j = D_j z_{ij} \tag{8}$$

then the system is capable of making a prediction, T_i, of its own input I_i, as in equation 6.9

$$T_i = \sum_j P_j D_j z_{ij} \tag{9}$$

This has been used to facilitate a noise filtering resonance algorithm with bidirectional weights z_{ij} and z_{ji} used to predict the input values. This resonance algorithm, developed from ideas of Grossberg [4], is crucial to the performance of the system in a noisy environment and has been detailed elsewhere [17]. In the normal case, without contamination from outlier data, the bidirectional weights should be equal ($z_{ij} = z_{ji}$).

The training algorithm, which minimises the squared error on the reconstructed input, follows naturally in equation 6.10

$$\Delta z_{ij} = k_j^{-1} P_j (I_i - T_i) \tag{10}$$

where k_j is given by equation 6.11

$$\Delta k_j = P_j (D_j - \beta k_j) \tag{11}$$

The term k ensures that learning proceeds as if forming a weighted mean of the examples of the input pattern with flexibility for change at a level determined by β. The number of extra presentations of data required to retrain part of the network is of the order $1/\beta$.

A fixed network architecture has a limit on information capacity. In an environment where the network is to be continually learning we must have ways of expanding this architecture by generating more nodes. The node-generation algorithm proposed here embodies the basic principle that in a purely unsupervised learning regime the density of nodes should be determined by the inherent precision of an input patterns (limited by the representations used), and *not* the within-class distribution of the training data sample as in other architectures. The generation rule is to generate a new node whenever $max(P_j)$ exceeds a given threshold ρ for any input pattern. For normalised input patterns, this implies that node generation may be triggered simply on the basis of node output.

This node-generation algorithm also ensures that all distinguishable patterns are encoded over several nodes. This is what is needed if the output of the network is to accurately reflect small changes in the input pattern. It is very similar, in principle, to the radial basis function approach suggested by [2] when the radial functions

[5]This is the temporal analogue to the spatial segmentation of data into different histogram bins. Recall that the system is not intended to learn dynamic patterns.

are normalised. This process has now been accepted as a standard method for improving the generalisation characteristics of interpolating systems [3].

The expected resolution of the system, embodied in the reproduceability of weight values δz_{ij}, is currently fixed. However, algorithms are being sought to allow these values to be adapted subject to negative reinforcement. The limiting accuracy is defined by the expected error on the inputs δI_i when $\delta z_{ij} = 0$. The weights to new nodes are initialised by the current input pattern, so that subsequent training only modifies these values slightly and the generated architectures can be used after a relatively short training period. This is to be compared with other neural architectures which initialise weights at random and thus take some time to settle down. Indeed such ideas as sparse connectivity and local representation over a few nodes, which emerge thanks to the node-generation algorithm, results in a network which trains in a time which is of linear order with respect to the number of input data patterns, as mentioned in the introduction.

During the previous study both the training algorithm and the node-generation algorithm were shown to converge to stable solutions for a fixed region of pattern space [17]. This stability is provided by both the training algorithm, and the increasing learning factor k_j. After a period of learning the outputs generated by the network for a particular input pattern thus become reproducible to within a known error limit and the training algorithm ceases to modify the weights.

5 Supervised Classification Layer

Supervised training of associative classification networks is generally done by node labeling, that is each node in the trained network is associated with a particular class type. There exist training algorithms for these systems which have been shown to be capable of approximating the performance of a Bayes classifier [10]. These classifiers are capable of returning the most likely class given the inputs but do not return any information on how likely this classification is to be wrong. Provided the classes are well separated in feature space, ambiguity of classification will not be a problem. However, as this situation is rare for real problems it would seem that current methods have limited applicability. With node labeling methods the decision boundaries are inherent in the network architecture and thus fixed for a particular classification task. A completely new architecture must be generated if the classification choices are re-specified for the same input data. This would be a limitation as a general learning environment particularly if a system were trying to develop its own classification scheme without prior access to a pre-defined set of classes. Thus, it is important for a classification scheme to be developed which preserves more information about the result of the classification process and is flexible enough to cope with modification of the classification scheme.

We can address first the information preservation aspect of the classification problem. Given the outputs of the CLAM network $P(j|I)$ (the probability that the input pattern I is consistent with the exemplar stored at each node j) information would be preserved if we were able to compute the conditional probability $P(C|I)$ that an input belonged to any externally defined class C. Other neural network architectures can be shown to approximate the same output response when trained with the appropriate algorithms [15] but do not have the internal structure to guarantee adequacy of the architecture to solve the task. In the case of CLAM however the result can be directly obtained as the following derivation demonstrates. Starting with equation 6.12:

$$P(C|I) = \sum_j P(C, j|I) \tag{12}$$

where j denotes a mutually exclusive data partition such as the selection of the nearest node. Then from Bayes theorem we can obtain equation 6.13

$$P(C|I) = \sum_j P(C|j, I)P(j|I) \tag{13}$$

but in a fully trained network, knowledge of I gives no additional information once j is known, so that $P(C|j, I) \approx P(C|j)$, giving equation 6.14

$$P(C|I) \approx \sum_j P(C|j)P(j|I) \tag{14}$$

This is a standard form for probability recoding by integrating over a marginal variable as suggested by [5] and used recently in other neural network architectures [12]. The method is directly comparable to the method of mixtures model classification used in the field of statistical pattern recognition [14]. Here we have the slight refinement in that $P(j|I)$ is defined for a locally computed[6] winner-take-all selection mechanism operating a statistically-based comparison metric for comparing frequency-coded signals.

[6] and neuronally plausible

Thus we need access to the quantity $P(C|j)$ which is the probability that the input pattern is of class C given that node j has been chosen as the maximum. For the output O_x from the classification node x, these connections can be trained using a running-average algorithm in equation 6.15

$$\Delta z_{jx} = k_x^{-1} P_j \alpha (C_x/\alpha - O_x) \tag{15}$$

where C_x is a measure of certainty of the classification result x which could either be a constant value (say 1) for a perfect supervisor or can be any independent scaled probability (for example $\alpha P(C|I)$ in the case of information provided by another classification system such as an associative network input stage). The factor α is a measure of the confidence of the supervision and ensures that training ceases if no information is available. The overall form of this training algorithm is distinctly Hebbian in the sense that reinforcement is largest when the activity o_j and O_x at both ends of the connection z_{jx} are large. The learning rate is controlled by k_x where this is determined in equation 6.16

$$\Delta k_x = \alpha P_j - \beta k_x \tag{16}$$

The β term again determines the steady state flexibility of learning for dealing with pattern variation. For the case $\beta = 0$ this equation provides a best estimate of $P(C|j)$ in the maximum likelihood sense and so can be regarded as optimal learning.

The overall probability can then be computed using an additional layer of neural architecture with distinct class nodes x connected to the intermediate mapping nodes j by weights z_{jx} proportional to $P(C|j)$. This is presented in figure 2 which shows both the unsupervised and the supervised layers[7].

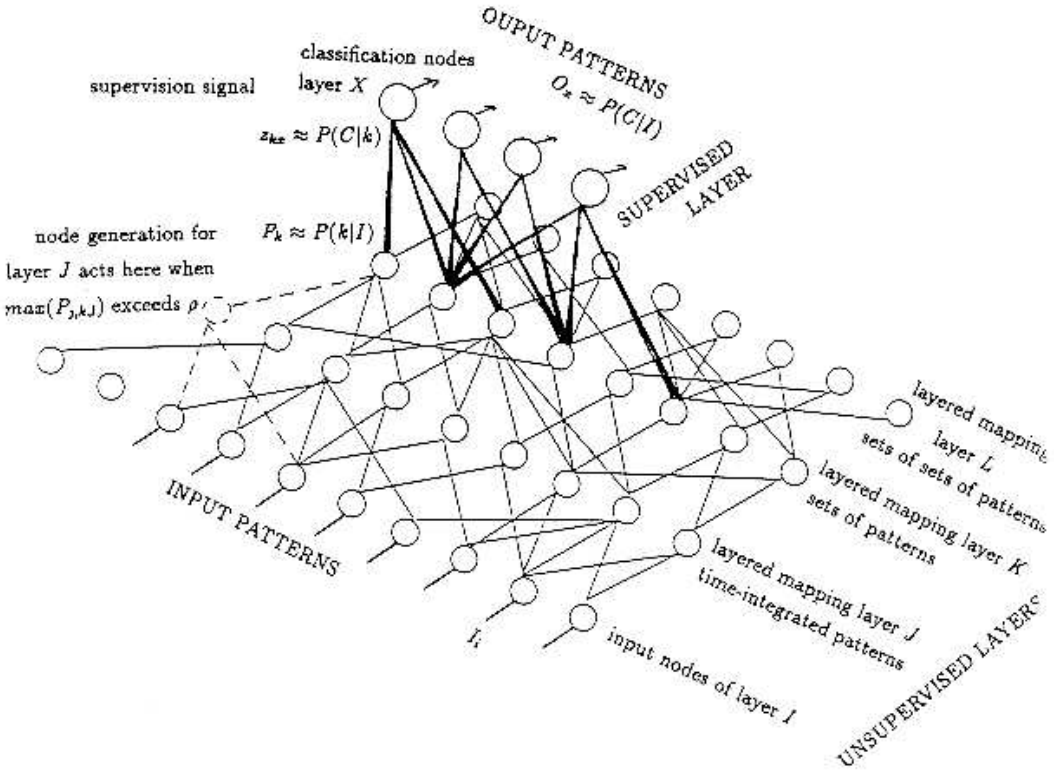

Figure 2: Probability Calculation

We then compute the output O_x from the node x using equation 6.17

$$O_x = \sum_j o_j z_{jx} \tag{17}$$

which will be proportional to the conditional probabilities that we seek, giving information about all possible classifications x.

[7] in actual fact this diagram shows the supervised layer X connected to the unsupervised layer K whereas in this discussion, we assume that layer X is connected to the preceding layer J. The layer selected depends on whether we wish to generate output probabilities for individual patterns or sets thereof.

As will be realised this supervised training algorithm required just one additional classification layer and can thus operate independently of the unsupervised training processes in the intermediate layer J. Connectivity between layers J and X can be sparse as we only need to create connections between those nodes in both layers which have correlated output activities (large $P(C|j)$). The intermediate association layer J must be stable so that the classification layer X can learn to compute the conditional probabilities. This property can be guaranteed by the CLAM algorithms.

The resulting hybrid classification system is capable of learning in both supervised and unsupervised as well as mixed supervision environments. It has been shown [13] that such architectures are capable of extremely rapid learning. This result is borne out in our experience.

6 Performance Test

For a given input, the network should provide an estimate of the conditional probability for a particular classification. In order to verify that the analytic approximation to the probability that the input pattern I is consistent with a template stored at a node in layer j, a comparison was carried out using a Monte-Carlo simulation. We also wish to demonstrate that it can cope with non-linearly separable and disjoint classes. Note that unsupervised learning and node-generation ability were demonstrated in the previous study. For an arbitrary data distribution, the only information available on which to base any estimates is the correspondence of input patterns with output classifications. Under these conditions, the best estimate for the conditional probability $P(C|I)$ is given simply by the ratio of the number of times that the particular input vector I has mapped to class C, divided by the total number of times that input vector I has occurred. It is therefore possible to determine the true probabilities, to any required accuracy, by means of a Monte-Carlo simulation. These can then be compared with the network outputs to evaluate performance and convergence properties.

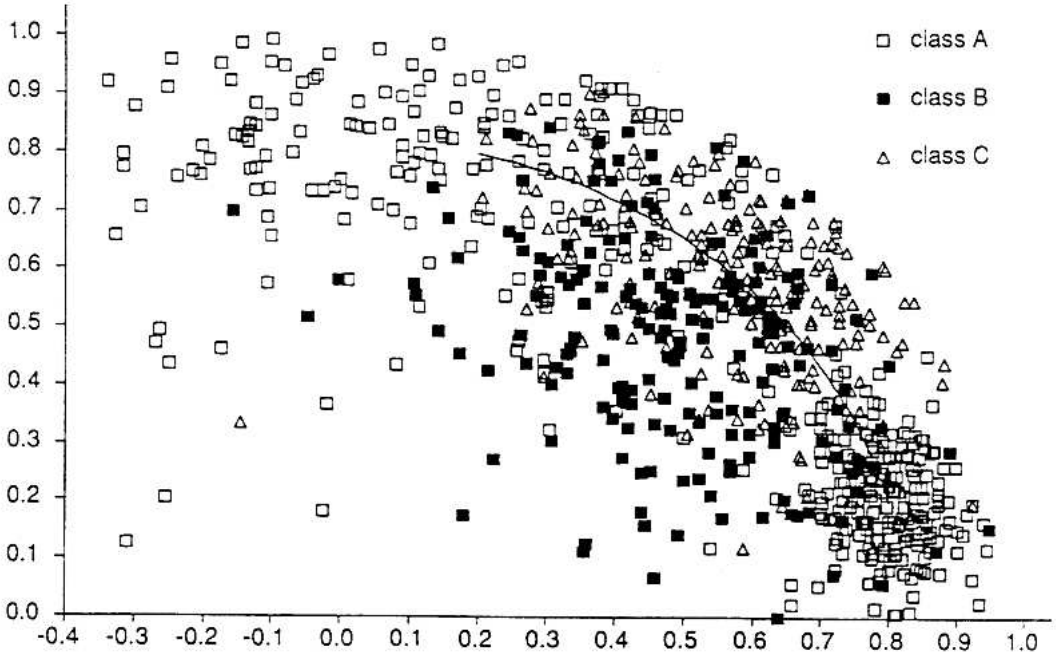

Figure 3: Class Occurences

The data for the Monte-Carlo simulation and training of the CLAM network is generated according a standard gaussian distribution[8]. The use of dot-products for the similarity measure in this network imposes certain constraints on the type of data we can use as input, and in particular it is necessary to generate normalised input vectors for the Monte-Carlo simulation if the results are to be compared with those of the network. In our test we have used an input data set generated as 3-element vectors representing xyz positions on the surface of a unit

[8]the use of gaussian distributions is not essential and clearly in principle any smoothly varying classification spaces could be mapped, however this apsect will not be demonstrated here.

radius sphere. This simple data set was partitioned into three classes according to location on the sphere. Random 3-element vectors I_1, I_2 and I_3 were then used as input to the first layer of the network.

Figure 3 shows a two-dimensional scatterplot projection of class occurrences. It shows that class A has a bi-modal distribution, with two overlapping classes, B and C, between its peaks. The expected probabilities are simply found by sampling the frequency of different class occurrences at a number of points along a trajectory in the input space (solid line in figure 5) (effectively a k-nearest-neighbour classifier). Figure 4 shows how intermediate nodes were positioned after training to map the input space to a resolution specified by the estimated error on the input components, parameters δz_{ij} and δI_i.

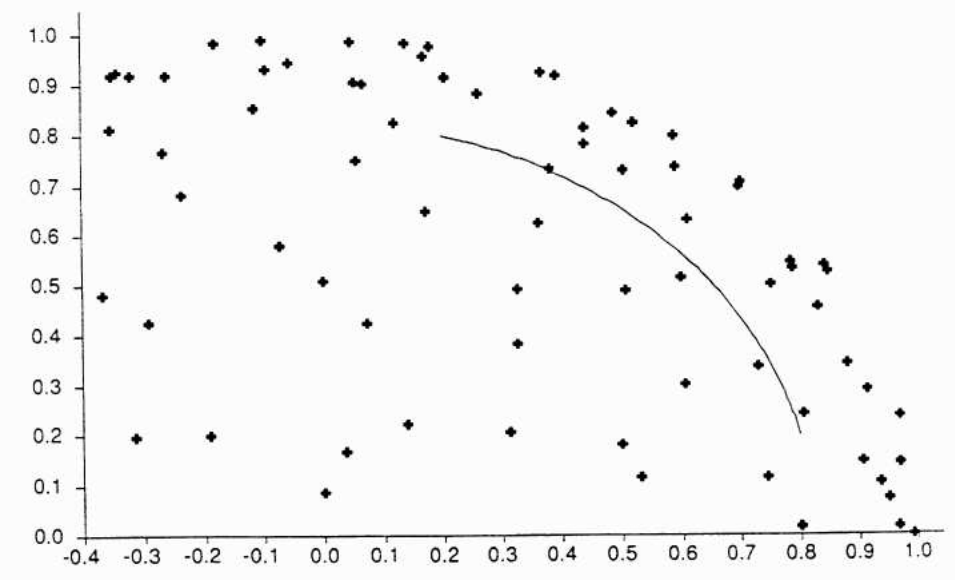

Figure 4: Node Positions

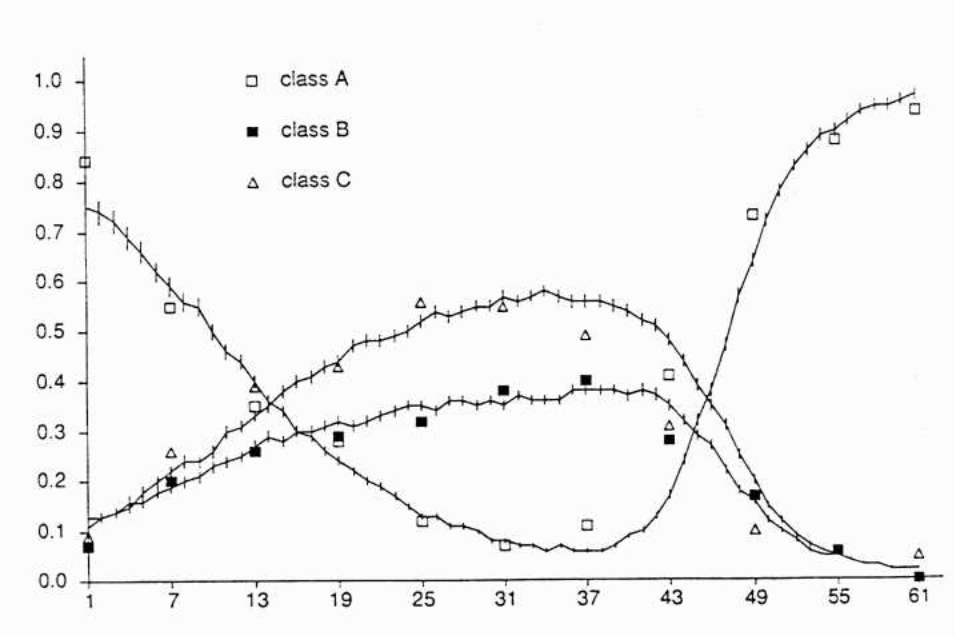

Figure 5: Network Output

To compare the CLAM estimates of the conditional probabilities with those of our Monte-Carlo simulation, we sampled the data at 61 points along the chosen trajectory through the input space. The output of the CLAM network is then found for 11 of these points (figure 5). The node outputs clearly model the correct probability density distributions of the input data. Optimal Bayesian classification is acheived simply by taking the maximum probability. For this purpose the new network will perform in a similar manner to more conventional node labelling

classification systems. Notice however that for this data set, class B would not have been represented by a conventional node labelling system as it is never the most likely classification, whereas in the CLAM architecture the full probability is generated for all classes, including class B. It therefore provides a more complete description of the input data for use by any further systems.

These probability estimations $P(j|I)$ [Appendix 2] were found to be accurate to within a few percent, in accordance with the specific choice of error models for δz_{ij} and δI_i. In this experiment a β of 0.01 was used. As expected, the network required relatively few training examples (100 examples of each class) to produce qualitatively correct results. Further training resulted in only a small improvement in accuracy. The results shown are for a network trained on 5,000 presentations of an example from each class. This compares with the 1,000,000 examples used to generate the Monte-Carlo distribution.

7 Conclusions and Future Work

In the work presented in this paper we have identified constraints which we believe should be considered when designing a self generating layered associative network. We recognise in particular the importance of information preservation as suggested by other authors [11] [16] and propose that in a layered network this can be achieved using a probabilistic interpretation of node output which can be related directly to frequency-coded signals operating a winner-take-all mechanism in a physiological system. We have used the principle of invariance on several occasions to constrain the algorithmic form of the network: input data representations are made invariant to the way that evidence towards a hypothesis is divided spatially amongst input nodes [Appendix 1]; and accumulated output data is made invariant to order of presentation and temporal segmentation of the data [Appendix 2]. The result is a set of algorithms that are a reasonable model of a stochastic neuronal system. In addition an attempt has been made to apply principles of uniformity and simplicity for instance in terms of number of layering types required[9].

These statistical assumptions restrict the application of the proposed network architecture to particular forms of input data, in particular frequency-coded signals (occurrence histograms) or discrete approximations to probability density functions. The use of input data in the form of frequency-coded histograms is particularly advantageous as it gives direct access to the expected variances and thus additional information about the input data. This kind of information is often regarded as being unavailable but is necessary for information preservation, error propagation and the probablistic interpretation of node function. It allows statistically-based decisions to be made regarding architecture generation and facilitates multi-layering.

A practical example of the use of histograms for the case of 2D object recognition and has been reported in [6] where shapes are represented as a histogram approximation for the probability density distribution of geometric co-occurrences. A theoretical analysis of the development of a self-generating classification system based on this input data has been reported in [7].

In this paper we have reported an extension to the CLAM architecture which supports the computation of an approximation to the true conditional probabilities of classification, so that more information is preserved for use by further processes. In addition this structure has flexibility, in that the classification probability for each class is independent of the other stored classes so new classification nodes can be added without disturbing the existing network structure. Bayesian decision performance can still be achieved by simply choosing the class with the highest probability but it can now also be done in the context of subsets of possible classes presented to the network during training. Such probabilistic information should enable the outputs of separate classification systems to be efficiently combined (see for example [1] to provide input to probabilistic inference systems and this is an area of research that we intend to explore.

Appendix 1

Given that we are to use the dot-product metric as the basis for node selection, we wish to show that the functional for the input to the network for input I_n at node n for an input of significance S given by equation 6.18

$$I_n = f(S) \tag{18}$$

can be uniquely defined by requiring that in a trained network the significance of a feature will have the same contribution to the selection of the maximum node irrespective of how it is divided amongst the inputs.

Consider one sub-feature S_o of the representation which initially resides entirely on one input node n. Its contribution to D_j is given by $\delta D_j = I_n z_{nj}$. Now imagine that this feature is translated so that it falls across

[9] such neurophysiologically-compatible principles makes the algorithms more biologically plausible.

the boundary of two inputs l and m with input values given by some function of the relative significance of each component a and b (such that $a + b = 1$). Then equation 6.19 holds.

$$I_l = f(S_o a) \text{ and } I_m = f(S_o b) \tag{19}$$

In order that correct account is now made of the combined input for this feature the total contribution to the indexing function for any other node k must be the same. This becomes equation 6.20

$$\delta D_j = f(S_o) z_{nj} = \delta D_k = f(S_o a) z_{lk} + f(S_o b) z_{mk} \tag{20}$$

As the network learns, z_{nj}, z_{lk} and z_{mk} will tend towards values proportional to their respective inputs. For a fully trained trained network where equation 6.21 holds,

$$z_{nj} \rightarrow f(S_o) \text{ , } z_{lk} \rightarrow f(S_o a) \text{ and } z_{mk} \rightarrow f(S_o b) \tag{21}$$

we can rewrite this equality as equation 6.22

$$f(S_o)^2 = f(S_o a)^2 + f(S_o b)^2 \tag{22}$$

and since $f(S)^2$ must be linear the only choice of function we can make is equation 6.23

$$f(S) = K S^{1/2} \tag{23}$$

Appendix 2

We wish to compute the probability that node $j1$ in layer j will be chosen in preference to node any other node. We start by computing the probability $P_{j1>j2}$ that the dot-product for node $j1$ will be greater than that for node $j2$. We define the difference $R_{j1>j2}$ in equation 6.24

$$R_{j1>j2} = D_{j1} - D_{j2} \tag{24}$$

Assuming that all errors on the input and weights are uncorrelated and normally distributed we can compute the variance on $R_{j1>j2}$ according to equation 6.25

$$\delta R^2_{j1>j2} = \sum_i (I_i^2 \times (\delta z^2_{ij1} + \delta z^2_{ij2}) + (z_{ij1} - z_{ij2})^2 \times \delta I_i^2) \tag{25}$$

Assuming that $R_{j1>j2}$ is normally distributed we get equation 6.26

$$P_{j1>j2} = \frac{1}{(2\pi.\delta R^2_{j1>j2})^{1/2}} \int_0^\infty e^{-(R_{j1>j2}-x)^2/2\delta R^2_{j1>j2}} dx \tag{26}$$

we now compute the probability that node $j1$ is the winner using the empirical relationship in equation 6.27

$$P_{j1} = \alpha \prod_j P_{j1>j} \tag{27}$$

where α is chosen so that equation 6.28 holds

$$\sum_j P_j = 1 \tag{28}$$

Appendix 3

We wish to show that the time integrated output from a layer can be made invariant to temporal segmentation of the input by suitable choice of the output function equation 6.29

$$o_j = P_j f(|I|) \tag{29}$$

The total significance of a pattern used to derive inputs I_i to the network is given by equation 6.30

$$S_{tot} = \sum_i I_i^2 = |I|^2 \tag{30}$$

Imagine that an input pattern is now segmented over time in the ratio $\tau : \omega$ so that the ratio of individual input components in preserved and the total input significance is the same as before so that we have equation 6.31

$$S_{tot} = \tau|I|^2 + \omega|I|^2 \tag{31}$$

Thus if the integrated output from layer j is to be invariant to this segmentation, equation 6.32

$$\sum_j o_j = \sum_j (o_j^\tau + o_j^\omega) \tag{32}$$

which can be written as equation 6.33

$$\sum_j P_j f(|I|) = \sum_j (P_j^\tau f(\tau^{1/2}|I|) + P_j^\omega f(\omega^{1/2}|I|)) \tag{33}$$

the probabilities are normalised, giving equation 6.34

$$f(|I|) = f(\tau^{1/2}|I|) + f(\omega^{1/2}|I|) \tag{34}$$

the only choice for the function f under these circumstances is given by equation 6.35

$$f(|I|) = K|I|^2 \tag{35}$$

Acknowledgements

The authors wish to thank Profs. John Mayhew and Peter Ivey for their support in developing this work, and the referees for helping us to achieve a clearer explanation of our work.

References

[1] Booth, D., Thacker, N.A., Pidcock, M.K. & Mayhew, J.E.W. (1991). Combining the Opinions of Several Early Vision Modules Using a Multi-Layered Perceptron, *Int. Journal of Neural Networks*, **2 (2/3/4)**, June-December, 75-79.

[2] Broomhead, D.S., & Lowe, D. (1988). Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. RSRE Memorandum 4148.

[3] Brown, M. & Harris, C. (1994) **Neurofuzzy Adaptive Modelling and Control**, Prentice Hall.

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

[5] Cox, R.T. (1946). *Am. J. Phys.*, **14**, 1-13.

[6] Evans, A.C., Thacker, N.A., & Mayhew, J.E.W. (1993). A Practical View Based 3D Object Recognition System. *IEE conference on neural networks*. Brighton.

[7] Evans, A.C., Thacker, N.A., & Mayhew, J.E.W. (1993). The Use of Geometric Histograms for Model-Based Object Recognition. *Proc. 4th BMVC93*. 429-438.

[8] Fukunaga, K. (1990). **Introduction to Statistical Pattern Recognition**, Academic Press Inc, second edition.

[9] Golden, R.M. (1988). A Unified Framework for Connectionist Systems. *Biological Cybernetics*, **59**, 109-120.

[10] Kohonen, T. (1988). The "Neural" Phonetic Typewriter. *IEEE Computer*, **11**, 22.

[11] Linsker, R. (1989). An Application of the Principle of Maximum Information Preservation to Linear Systems. *Advances in Neural Information Processing Systems*, **1**, 186-194.

[12] Lutterell, S.P. (1993). An Adaptive Bayesian Network for Low level Image Processing. *IEE conference on neural networks*. Brighton.

[13] Moody, J. & Darken, C.J. (1989). Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*, **1**, 281-294.

[14] Pudil, P., Novavicova, J., & Kittler, J. (1993). Automatic Machine Learning of Decision Rules for Classification Problems in Image Analysis, *Proc. 4th BMVC93*, **1**, 15-25.

[15] Richard, M.D., & Lippmann, R.P. (1991). Neural Network Classifiers Estimate Bayesian a posterioi Probabilities. *Neural Computation*, **3**, 461-483.

[16] Sanger, T.A. (1989). An Optimality Principle for Unsupervised Learning. *Advances in Neural Information Processing Systems*, **1**, 11-20.

[17] Thacker, N.A. & Mayhew, J.E.W. (1989). Designing a Network for Context Sensitive Pattern Classification. *Neural Networks*, **3**, 3.