A PRINCIPLED APPROACH TO *N*-TUPLE RECOGNITION SYSTEMS

N M Allinson and A Kolcz

1 Introduction

The idea of *n*-tuple sampling as a basis for pattern recognition was first proposed in 1959 by Bledsoe and Browning [1]; and remains a viable approach to a range of pattern classification tasks especially where speed of learning is of importance. Though a firm understanding of their operation was developed in these early years and is summarised in Ullmann's Pattern Recognition Techniques [2], only recently has the formal relationship between *n*-tuple neural networks (as they have become to be called) and more mainstream network paradigms, such as radial basis function networks, and classical non-parametric pattern classifiers, such as kernel estimation, been presented [3, 4]. This contribution describes how the classic *n*-tuple recogniser and the more recently developed n-tuple regression network form differing approximations in the classification process.

The essential elements of the classic *n*-tuple recogniser are shown in Figure 1. The input data is projected on to a two-dimensional binary retina (The retina is not an essential feature but remains in most descriptions for historical reasons); and this retinal pattern is sampled by taking fixed randomly-positioned chosen array bits – *n* at a time. The state of these *n*-tuples form the addresses of a number of memory nodes. These memory states after training can be considered as estimating the probability of occurrence of the individual tuple states. In traditional *n*-tuple systems, the memory contents are either a simple class label that indicates that this located has been addressed or a counter that indicates how often the particular address has been activated. Here, we will concentrate on this latter variant – the frequency *n*-tuple recogniser, first proposed in [5] – and the recently suggested *n*-tuple regression network [6], which was originally presented as a function approximator.

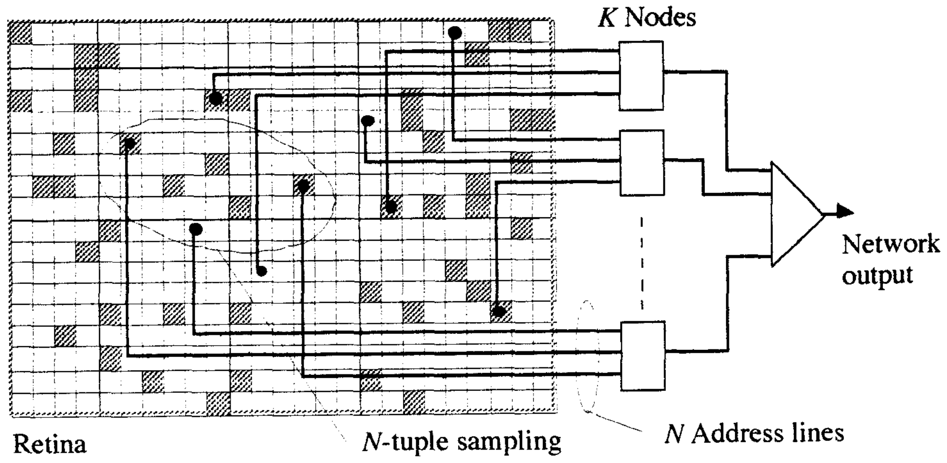

Figure 1 *Basic structure of an n-tuple network*

Department of Electrical Engineering and Electronics, UMIST, PO Box 88, Manchester, M60 1QD, UK

2 Maximum likelihood approximation

This section owns much to the analysis given by Ullmann [2]. The training phase of a supervised network provides estimates of the conditional probabilities of individual pattern classes. The *class membership probabilities* can be formulated through the Bayes relationship, i.e.

$$P(\mathbf{x} \in c) = \frac{P(\mathbf{x}|c)P(c)}{P(\mathbf{x})} \tag{1}$$

where c is the *class label* for a particular class $\{c = 1,2,\ldots,C\}$.

For an n-tuple network, the *prior probabilities and conditional probability densities* can be expressed through the individual tuple terms, if we assume the statistical mutual independence of the samples. That is

$$P(\mathbf{x}) \cong \hat{P}(\mathbf{x}) = P(t_1(\mathbf{x})) \cdot P(t_2(\mathbf{x})) \cdot \ldots \cdot P(t_K(\mathbf{x})) \tag{2}$$

$$P(\mathbf{x}|c) \cong \hat{P}(\mathbf{x}|c) = P(t_1(\mathbf{x}|c)) \cdot P(t_2(\mathbf{x}|c)) \cdot \ldots \cdot P(t_K(\mathbf{x}|c)).$$

These tuple-wise probabilities can be, in turn, estimated by considering the frequency each tuple location has been addressed during training. Namely,

$$P(t_k(\mathbf{x})) \cong \frac{a_k}{T} \tag{3}$$

$$P(t_k(\mathbf{x}|c)) \cong \frac{w_k^c}{T_c}$$

and $P(c) \cong \frac{T_c}{T}$.

where T_c is the number of input patterns for class c and T is the total number of patterns. Hence, the *posterior probability* for class membership is given by

$$P(\mathbf{x} \in c) \cong P(c) \prod_{k=1}^{K} \frac{P(t_k|c)}{P(t_k)} = \frac{T_c}{T} \prod_{k=1}^{K} \frac{T \cdot w_k^c}{T \cdot a_k} = \left(\frac{T_c}{T}\right)^{1-K} \prod_{k=1}^{K} \frac{w_k^c}{a_k}. \tag{4}$$

A problem with this relationship is when one or more tuple locations have not been addressed during the training phase (i.e., $w_k^c = 0$ or $a_k = 0$). Several *ad hoc* solutions have been suggested from simply ignoring these unaddressed terms from the product to inserting small numeric values [7]. For simple classification, the detailed values of these estimated probabilities are unimportant as only their mutual relationship matters. The above expression can be simplified by removing common factors to yield the following decision rule for selecting the winning class c_{winner}.

$$c_{\text{winner}} = \max_{c=1,\ldots,C} T_c^{K-1} \prod_{k=1}^{K} w_k^c \tag{5}$$

If the number of training examples per class is a constant, then further simplification yields,

$$c_{\text{winner}} = \max_{c=1,\ldots,C} \prod_{k=1}^{K} w_k^c . \tag{6}$$

The validity of this approach depends on the assumed mutual independence of the tuples. With random sampling without repetitions, this may be justified.

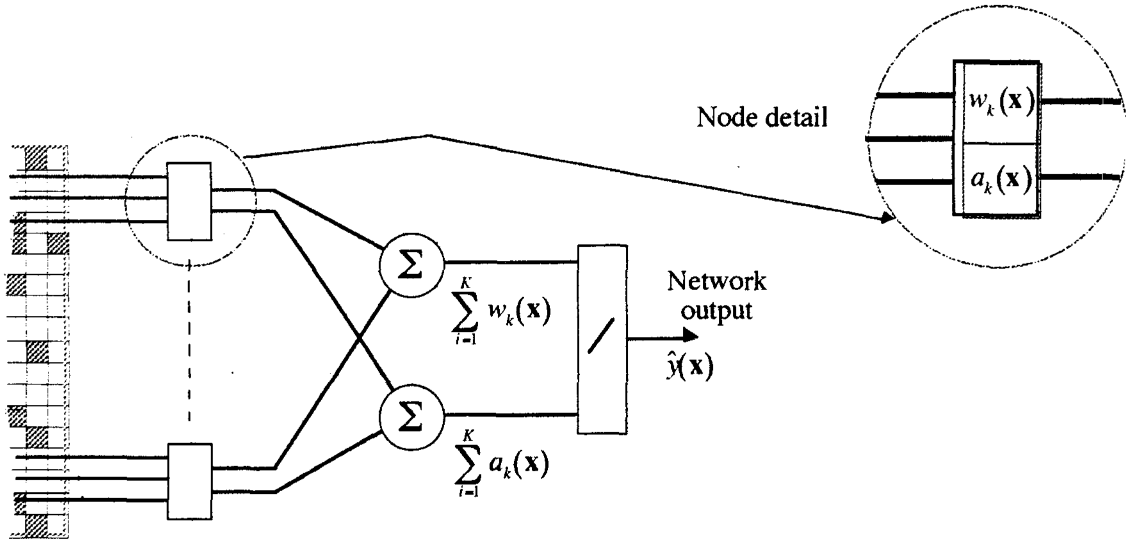

Figure 2 N-tuple regression network – output stage modifications

3 Regression approximation

The n-tuple regression network was originally presented as a function approximator, that is the task of estimating a function, $f = (\mathbf{x})$, given a finite number of sample pairs $(\mathbf{x}_i, y_i)$. The network is a simple extension of the frequency n-tuple network where the node memories contain not only a counter value, a_k, indicating the number of times the location k has been addressed but also an associated weight value, w_k. The modifications to the conventional n-tuple scheme are indicated in Figure 2. During training, each addressed tuple location is updated using

$$w_k(\mathbf{x}_i) \leftarrow w_k(\mathbf{x}_i) + y_i \quad \text{and} \quad a_k(\mathbf{x}_i) \leftarrow a_k(\mathbf{x}_i) + 1 . \tag{7}$$

Initially, all weight and counter values are set to zero. After training the network output, $\hat{y}(\mathbf{x})$, is given by

$$\hat{y}(\mathbf{x}) = \frac{\sum_{k=1}^{K} w_k(\mathbf{x})}{\sum_{k=1}^{K} a_k(\mathbf{x})} . \tag{8}$$

This response, $\hat{y}(\mathbf{x})$, has been shown to be the best estimate of the underlying function for $y(\mathbf{x})$ [6] and can formally be expressed in terms of the *distance metrics* resulting from the number of different n-tuple addresses generated for two arbitrary inputs $\mathbf{x}'$ and $\mathbf{x}''$, and the Hamming distance $H(\mathbf{x}', \mathbf{x}'')$ – that is the number of bits for which $\mathbf{x}'$ and $\mathbf{x}''$ differ. (In terms of function approximation, then it is usual to consider *kernel widths* rather than distance metrics.) The expected values of the distance $\rho(\mathbf{x}',\mathbf{x}'')\ \{\equiv \rho(H)\}$ can be approximated by

$$E\{\rho(\mathbf{x}',\mathbf{x}'')\} = K\left(1-\exp\left(-H/_K\right)\right), \tag{9}$$

for tuple sampling without repetitions.

and

$$E\{\rho(\mathbf{x}',\mathbf{x}'')\} = K\left(1-\exp\left(-NH/_R\right)\right), \tag{10}$$

for tuple sampling with repetitions and where R is the size of the retina in bits. So the inherent distance metric for n-tuple sampling is seen to be an increasing monotonic function. Derivation and further details are given in [8].

Returning to the expression for the network response then, as derived in [6], the best estimate of the underlying regression function is given by

$$\hat{y}(\mathbf{x}) = \frac{\sum_{i=1}^{T} y_i \left\{1-\frac{\rho(\mathbf{x},\mathbf{x}_i)}{K}\right\}}{\sum_{i=1}^{T}\left\{1-\frac{\rho(\mathbf{x},\mathbf{x}_i)}{K}\right\}}. \tag{11}$$

We can simply reformulate this expression in terms of a pattern classification task into C distinct classes. The network through training approximates C *indicator functions*, each one of which denotes membership of an individual class. Namely

$$I_c(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \subset c \\ 0 & \text{otherwise} \end{cases}. \tag{12}$$

These indicator functions can be approximated, after training, by

$$I_c(\mathbf{x}) = \frac{\sum_{i=1}^{T}(\mathbf{x}^i \subset c)\left(1-\rho(\mathbf{x},\mathbf{x}^i)/K\right)}{\sum_{i=1}^{T}\left(1-\rho(\mathbf{x},\mathbf{x}^i)/K\right)} = \frac{\sum_{k=1}^{K} w_k^c}{\sum_{k=1}^{K} a_k} \equiv P(c)\frac{\sum_{k=1}^{K} P(t_k|c)}{\sum_{k=1}^{K} P(t_k)}. \tag{13}$$

This relationship gives the ratio of the cumulative summation of all training points belonging to class c, which have an n-tuple distance at $0,1,\ldots,(K-1)$ from $\mathbf{x}$ to a similar cumulative summation for all training points. Ignoring common terms, the winning class is simply given by

$$c_{\text{winner}} = \max_{c=1,\ldots,C} \sum_{k=1}^{K} w_k^c. \tag{14}$$

The mutual exclusion of individual tuple samples is not required to ensure validity of this second approach. In fact, oversampling can lead to greater consistency in the formulation of tuple distances. The presence of unselected locations is not a problem and the product of terms is replaced by the more computationally effective sum of terms. For classification tasks, only integer arithmetic is necessary as the network targets can only take the value 0 or 1.

4 Comparison of approaches

Both the maximum likelihood and the regression methods aim to provide a good approximation to the underlying true regression function; but they result in different computational forms. A question that naturally arises is which method is more accurate and under what conditions will they yield identical results. For equal occurrence of training classes, the decision provided by both methods will be in agreement when,

$$\prod_{k=1}^{K} w_k^a \geq \prod_{k=1}^{K} w_k^b \Leftrightarrow \sum_{k=1}^{K} w_k^a \geq \sum_{k=1}^{K} w_k^b, \tag{15}$$

where the indices a and b correspond to the selection of different classes. This relationship is valid if the weights in the two competing selections can be rank ordered, that is

$$w_k^a \geq w_k^b \qquad \text{for } k = 1,2,\ldots,K.$$

Otherwise, the output decisions may differ and will depend on the specific distribution of the weight values for all competing classes.

The nature of the decision surfaces implicit in each approach can been seen by noting that the selection of a set of K weights is equivalent to selecting a point in $\Re^K$ space. As all weights are positive, only a portion of this space needs to be considered. Selecting a point (i.e., a set of weights) determines the corresponding decision surfaces. Namely,

$$\sum_{k=1}^{K} w_k = \text{constant} \qquad \text{and} \qquad \prod_{k=1}^{K} w_k = \text{constant}.$$

The former, for the regression approximation, represents a hyperplane; while the latter, for the maximum likelihood approximation, represents a hyperboloid in the $\Re^K$ space. Hence the regions of $\Re^K$ yielding a consistent decision differ for the two approximations. For an idealised $\Re^2$ space (Figure 3), the regions where decisions will differ are shown shaded for a situation where the point (4, 1) has been selected. From this example, it can be inferred that the regression network will favour situations where there is an unbalanced weight distribution (i.e., some tuples provide a high response and others low); whereas the ML network will emphasise classes where the majority of tuples have similar (not necessarily high) responses. In this example, the ML network would choose the point (2, 2.5) while the regression network would choose (4, 1).

5 Managing addressed null weights

As discussed previously, the null weights selected during a ML recall operation are often assigned an arbitrary fractional value. An alternative suggestion would be to set all weights to unity prior to training (or equivalently incrementing all weights by unity prior to forming the product terms). There would be a computational advantage in this approach as only integer arithmetic is necessary. Considering, as before, the decision hypersurfaces it is possible to compare the classification performance with the traditional approach. The two methods will be consistent for weight space regions, for two nominal pattern classes *c* and *d*, where

$$\prod_{k=1}^{K} w_k^c \geq \prod_{k=1}^{K} w_k^d \Leftrightarrow \prod_{k=1}^{K}\left(w_k^c+1\right) \geq \prod_{k=1}^{K}\left(w_k^d+1\right). \tag{16}$$

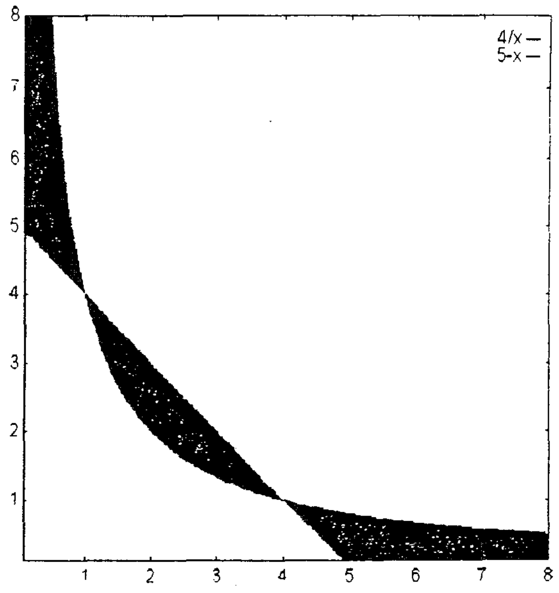

Figure 3 Decision surfaces in $\Re^2$ space for ML and regression approximations

If a particular point in the weight space is chosen (say, $w_1^c,\ldots,w_K^c$) then the region where decisions will differ is delimited by the two hyperboloid surfaces passing through this point. Namely,

$$\prod_{k=1}^{K} w_k = \prod_{k=1}^{K} w_k^c \quad \text{and} \quad \prod_{k=1}^{K}\left(w_k+1\right) = \prod_{k=1}^{K}\left(w_k^c+1\right). \tag{17}$$

Figure 4 illustrates this situation for an idealised $\Re^2$ space. The basic hyperbola satisfies the equation $w_1 \cdot w_2 = 100$ and the reference points are chosen as (10, 10) and (20, 5) for the upper and lower graphs respectively. The straight line corresponds to the regression network decision boundary. The region of decision disagreement between the original and modified ML methods is very small; though in this example the new method yields a decision only slightly biased in the direction of the regression network decision.

6 Simulation experiments

Though the usefulness or otherwise of any recognition system can only be confirmed by extensive application under practical conditions, the potential of a particular scheme (and its meaningful comparison to others) is best tackled through controlled simulation. For these comparative experiments, the input retina is a 16 x 16 binary array consistent with input pattern classes of simple geometric shapes – ten classes in total with a black (1) : white (0) pixel ratio varying from 50% to 25%. The data set

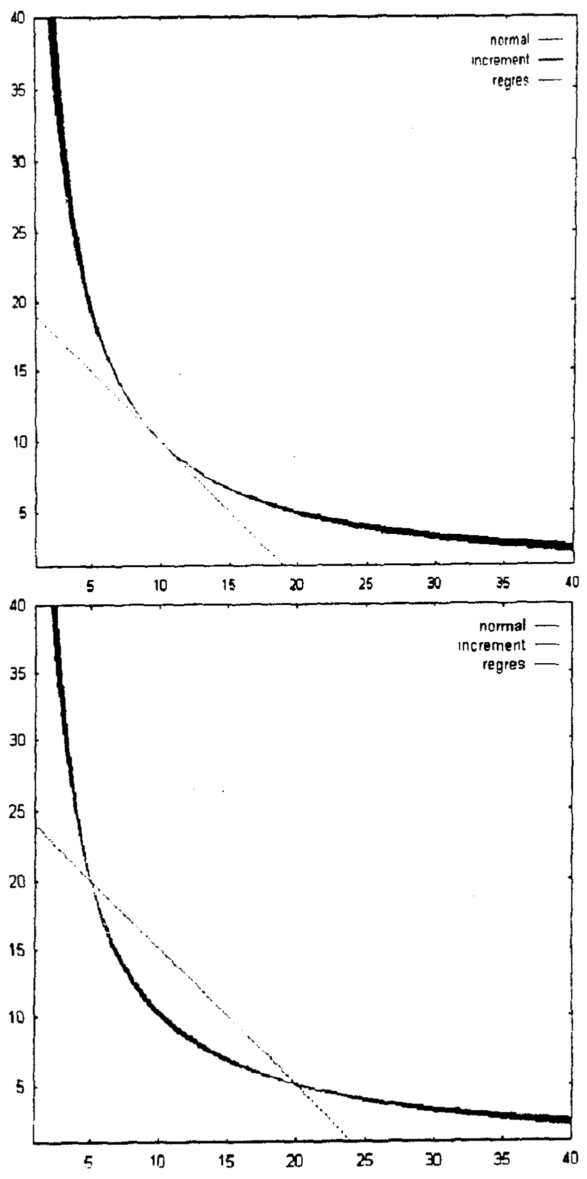

Figure 4 *Decision surfaces in $\Re^2$ space for classic and modified frequency n-tuple networks The corresponding regression approximation is shown as a straight line.*

consists of these ten pattern classes with varying degrees of additional random noise (probability, p, of an individual pixel being correctly set of $p = 0.85$, 0.75 or 0.65). Examples of the data set are given in Figure 5. The training set composes 20 or 40 blocks of these pattern classes, while the test set consists of 80 blocks.

The n-tuples are chosen randomly with n varying between 3 and 31 (in unit steps). The number of selected tuples was kept constant such that the retina is fully sampled at the minimum tuple size. Hence for larger tuples, the input retina is oversampled. Though this chosen procedure is rather arbitrary, it does improve the quality of the tuple distance function approximation as the tuple size increases. Otherwise the fraction of tuple memories addressed during training reduces as the tuple size increases so the recall operation is based on diminishingly small numbers (i.e., the statistical consistency begins to

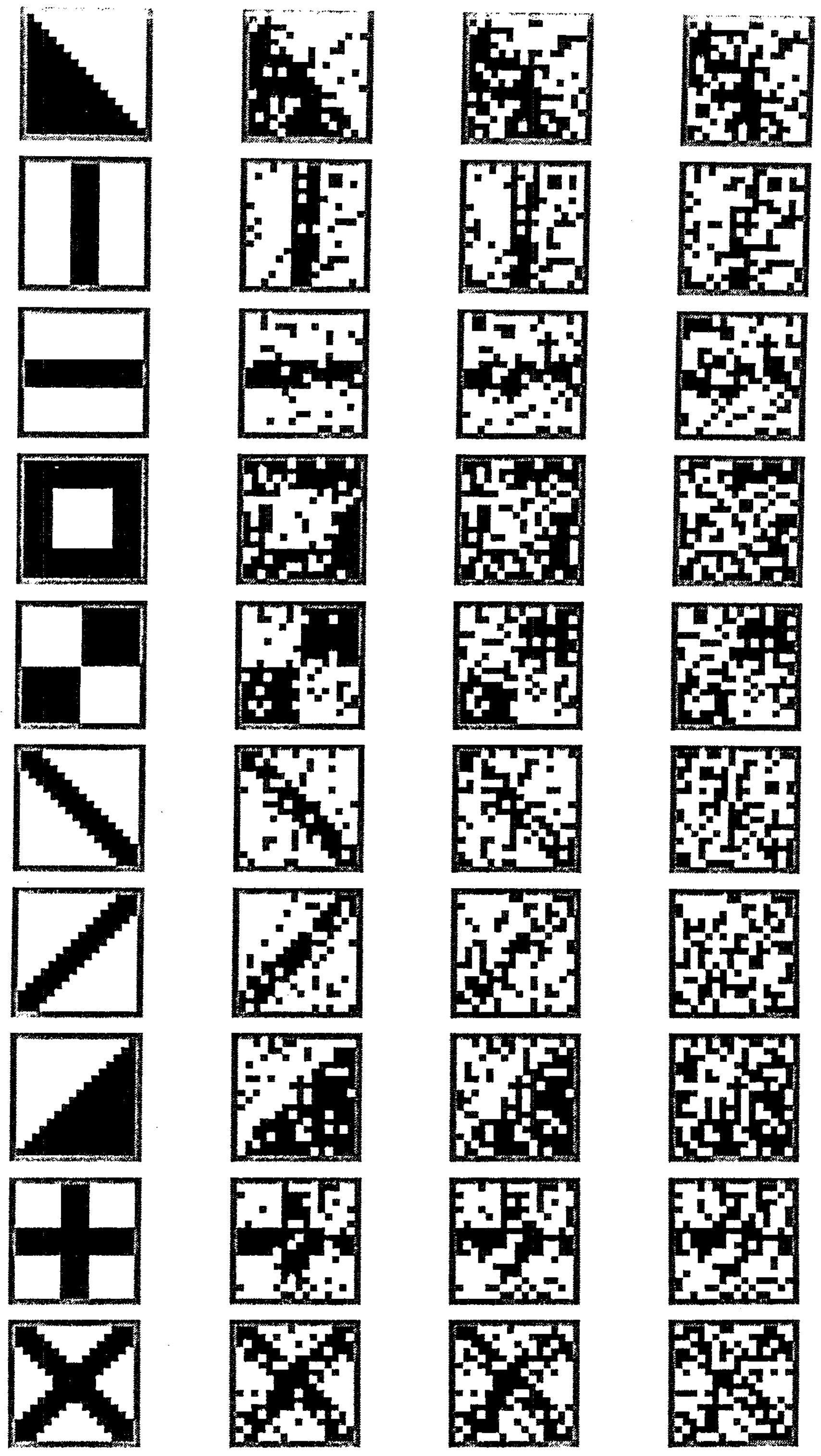

Figure 5 *Simulation training data*
The second, third and fourth columns are examples of the training data for p = 0.85, 0.75 and 0.65 respectively

become suspect). The solution is either to increase the size of the training set (not always feasible in practice) or increase the number of tuple samples.

Table 1 shows the averaged correct classification rate (over eight simulation runs), with 20 training blocks and the test blocks being independent of the training ones, for the following training schedules:

1. Frequency n-tuple network (ML decision with null weights ignored)
2. Frequency n-tuple network (ML decision with null weights set to 0.5)
3. Frequency n-tuple network (ML decision with null weights set to 1.0)
4. Regression n-tuple network.

Though all approaches achieve almost identical recognition for small n, the regression network maintains a higher classification rate as n increases above this optimum region. This is because for large n, anincreasing fraction of locations will be empty (or contain only unity values) – which are treated indiscriminately by the ML approach. The regression network also provides superior performance as the training/test data becomes noisier (i.e., p decreases). As p approaches 0.5, the patterns become increasingly random and the interclass variability diminishes.

Table 1 Classification rates

	$p = 0.65$				$p = 0.75$				$p = 0.85$			
$n =$	4	8	16	31	4	8	16	31	4	8	16	31
ML net (null weights ignored)	.82	.53	.02	0	.98	.95	.19	0	1.00	.99	.70	.21
ML net (null weights = 0.5)	.82	.67	.33	.21	.98	.98	.63	.21	1.00	1.00	.97	.39
ML net (null weights = 1.0)	.81	.68	.33	.21	.98	.98	.63	.21	1.00	1.00	.97	.39
Regression net	.82	.70	.34	.21	1.00	1.00	.96	.06	1.00	1.00	.99	.39

Variants of the frequency n-tuple network where the null weights are set to some non-zero value display a large improvement over the classic approach where null weights are ignored – this is particularly true for regions of non-optimal n. The regression network still provides better performance albeit sometimes only marginally. Figure 6 shows a typical set of averaged simulation results for the regression network and the classic frequency n-tuple network. For these simulations, the optimum tuple size is generally small (typically $n = 3 - 6$) with a fall-off in performance as n increases. This observation is due to the nature of the training patterns (i.e., approximately equispaced in the pattern space and with high noise levels). In general, there will be a peak in performance as n increases with the peak being wider and less distinct for the regression network than the other approaches (i.e., optimisation of this network is less critical).

7 Conclusion

This short paper has shown how the regression n-tuple network, originally developed for function approximation, can be employed in recogniser systems. This network and the traditional frequency n-tuple networks make approximations to evaluate the underlying regression function associated with the individual pattern classes. The regression network approximation does not require the assumption of the mutual independence of tuple samples and it yields a computationally simpler form. A minor improvement in the management of addressed null weights for frequency n-tuple networks has also been presented. This, too, yields a computational simplification. An extensive programme of simulations have supported these statements.

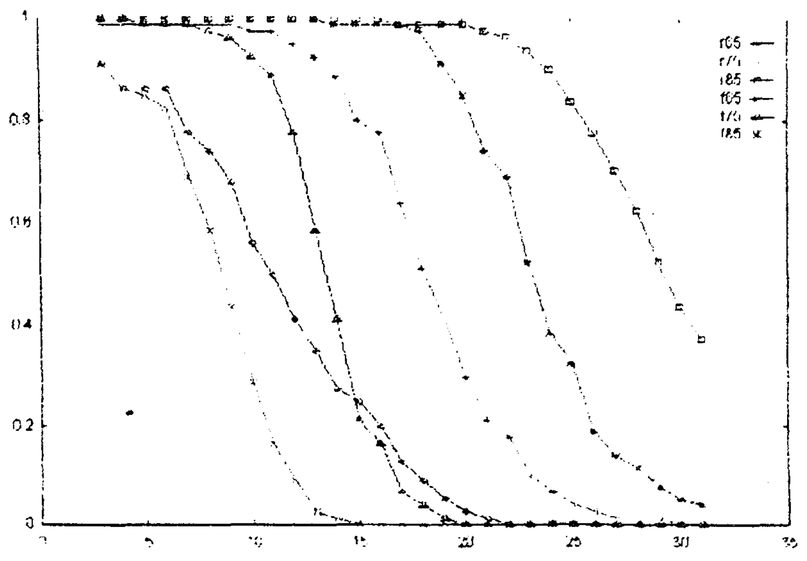

Figure 6 *Set of averaged classification results for regression network and classic n-tuple frequency network*

8 References

[1] W W Bledsoe and I Browning (1959), *Pattern recognition and reading by machine*, IRE Joint Computer Conference, 225-232

[2] J R Ullmann (1973) Pattern Recognition Techniques, London: Butterworth

[3] A Kolcz and N M Allinson (1995), *General Memory Neural Network – extending the properties of basis networks to RAM-based architectures,* 1995 IEEE International Conference on Neural Networks, Perth, Western Australia, 1638-1643

[4] A Kolcz and N M Allinson (in press), *The General Memory Neural Network and its relationship with* basis *function architectures*, IEEE Transactions on Neural Networks

[5] W W Bledsoe and C L Bisson (1962), *Improved memory matrices for the n-tuple pattern recognition method*, IRE Transactions on Electronic Computers, **11**, 414-415

[6] A Kolcz and N M Allinson (1996), *N-tuple regression network*, Neural Networks, **9**, 855-869

[7] N M Allinson and A R Kolcz (in press), *N-tuple neural networks*, Annals of Mathematics in Artificial Intelligence

[8] N M Allinson and A Kolcz (1995), *The theory and practice of n-tuple neural networks*, in Neural Networks (Taylor J G, ed.), Alfred Waller, 53-70

oOo