The Theoretical and Experimental Status of the n-tuple Classifier

Article *in* Neural networks: the official journal of the International Neural Network Society · February 1998

DOI: 10.1016/S0893-6080(97)00062-2 · Source: PubMed

2 authors, including:

Richard Rohwer
SRI International
62 PUBLICATIONS **751** CITATIONS

SEE PROFILE

The Theoretical and Experimental Status of the n-tuple Classifier

Richard Rohwer and Michał Morciniec

Dept. of Computer Science and Applied Mathematics

Aston University

Birmingham, UK B4 7ET

July 11, 1996

corresponding author:

Dr Richard Rohwer

Dept. of Computer Science and Applied Mathematics

Aston University, Birmingham, UK B4 7ET

e-mail: rohwerrj@cs.aston.ac.uk

phone: 0121 359 3611 x.4688

fax: 0121 333 6215

Abstract

A number of theoretical approaches related to the n-tuple classification system are reviewed including Kanerva's SDM, the n-tuple regression network, the Hamming–Distance framework and Likelihood Estimation. The limitations of these methods are pointed out and resemblances that exist between them are underlined. Large scale experiments carried out on StatLog project datasets confirm the n-tuple method as a viable competitor to more popular methods due to its speed, simplicity, and accuracy on the majority of a wide variety of classification problems. A further investigation into the failure of the method on certain datasets shows its inner workings and turns up two main problems: difficulties with highly skewed class priors and more importantly, a mismatch between the scales involved in generalisation, the amount of training data available, and the volume of the region in which data is likely to exist. This highlights areas where improvements in the method are needed and further theoretical progress would be helpful.

keywords: n-tuple classifier, Hamming distance, regression network, Kanerva model, CMAC code, review, benchmarking, StatLog.

1 Introduction

The n-tuple classifier, invented by Bledsoe and Browning in 1959 (Bledsoe & Browning, 1959), is one of the oldest practical pattern recognition methods based on distributed computation and amenable to description in terms of neural network metaphors. Although eclipsed in popularity by methods such as Multilayer Perceptrons and Radial Basis Function networks, the n-tuple method continues to offer properties which make it vastly superior for certain common purposes. First among these properties is its speed of operation. The training algorithm is a one-shot memorisation task, computationally trivial compared to solving linear systems or minimising nonlinear functions. Another advantage is the sheer simplicity of learning by memorisation. If this can form the basis of a sound pattern recognition principle, then it is arguable that biological systems could make use of it.

It is prudent to suspect that relatively poor performance will accompany the speed and simplicity of the n-tuple algorithm. There are many reports of satisfactory results with the method (Aleksander & Stonham, sh1979a; Bledsoe & Bisson, 1962; Ullmann & Kidd, 1969; Ullmann, 1969; Tarling & Rohwer, 1993; Rohwer & Lamb, 1993) but few studies involving comparisons with other methods (Rohwer & Cressy, 1989). Furthermore, most studies use just one or two small data sets. Therefore a large experiment was carried out, in which the n-tuple method was tested on 11 large real-world data sets which had been previously used by the European Community ESPRIT StatLog project (Michie *et al.*, 1994) to test 23 other classification algorithms including the most popular neural network methods. The results, presented in section 5.4, show no systematic performance gap between the n-tuple method and the others, on 7 of the 11 data sets tested. It is easy to recognise the other 4, without referring to any competing methods, because the n-tuple method performed no better than random guessing. The experiments therefore suggest that in most cases the n-tuple method

gives competitive performance, and the cases when it does not are clearly recognisable.

When a fast and simple method proves to be a competitive performer in a large set of experiments, one would like to know why. Unfortunately, the n-tuple method has not yet yielded to theoretical analysis as well as the optimisation-based approaches which can be embedded in Bayesian statistical theory (MacKay, 1992) but there has been some progress. The main theoretical results are reviewed, discussed, and used as a vehicle for explaining some relationships between the standard n-tuple method, n-tuple methods incorporating linear regression, radial basis function networks, and the Kanerva associative memory model. The theoretical tools are then used to develop a semi-quantitative account of why the method failed on 4 of the 11 datasets. The main theoretical stumbling blocks are also indicated.

2 The n-tuple recognition method

The n-tuple recognition method is also known as a type of "RAMnet"[1] or "weightless neural network". It forms the basis of a commercial product (Aleksander *et al.*, 1984). It is a method for classifying binary patterns, which can be regarded as bit strings of some fixed length L. This is not an important restriction, because there is an efficient preprocessing method, tailored to the RAMnet's generalisation properties, for converting scalar attributes into bit strings. This method is reviewed in section 4. It is also possible to generalise a special case of the n-tuple method from a classifier to a function interpolator.

[1] *RAMnet*s also include stochastic generalisations, *pRAM*s, to which the n-tuple recognition algorithm is not applied. These are not considered here.

2.1 Definition of the standard n-tuple method

A consise definition of the n-tuple method is given here. More descriptive definitions can be found in (Alexander & Morton, 1995; Aleksander & Stonham, 1979b). Several (let us say N) sets of n distinct[2] bit locations are selected randomly. These are the *n-tuples*. Collectively, they are called the "*input mapping*." The restriction of a pattern to an n-tuple can be regarded as an n-bit number which, together with the identity of the n-tuple, constitutes a "feature" of the pattern. The standard n-tuple recogniser operates simply as follows:

> *A pattern is classified as belonging to the class for which it has the most features in common with at least 1 training pattern of that class.* (1)

This is the $\theta = 1$ case of a more general rule whereby the class assigned to unclassified pattern $\boldsymbol{u}$ is

$$\underset{c}{\operatorname{argmax}} \left(\sum_{i=1}^{N} \Theta_\theta \left(\sum_{\boldsymbol{v} \in \mathcal{D}_c} \delta_{\alpha_i(\boldsymbol{u}),\alpha_i(\boldsymbol{v})} \right) \right) \tag{2}$$

where $\boldsymbol{v}$ ranges over $\mathcal{D}_c$, the set of training patterns in class c, $\Theta_\theta(x) = x$ for $0 \leq x \leq \theta$, $\Theta_\theta(x) = \theta$ for $x > \theta$, $\delta_{i,j}$ is the Kronecker delta[3] ($\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.) and $\alpha_i(\boldsymbol{u})$ is the $i^{\underline{\text{th}}}$ feature of pattern $\boldsymbol{u}$:

$$\alpha_i(\boldsymbol{u}) = \sum_{j=0}^{n-1} u_{\eta_i(j)} 2^j. \tag{3}$$

Here u_k is the $k^{\underline{\text{th}}}$ bit of $\boldsymbol{u}$ and $\eta_i(j)$ is the $j^{\underline{\text{th}}}$ bit location of the $i^{\underline{\text{th}}}$ n-tuple.

Small values of θ, greater than 1, are sometimes found to be most effective (see section 3.3).

With C classes to distinguish, the system can be implemented as a network of NC nodes, each of which is a 1-bit random access memory (RAM); hence the term *RAMnet*.

[2] Relaxing the requirement that an n-tuple has n *different* bit locations amounts to introducing a mixture of differently sized n-tuples. Note the restriction does not disallow a single pattern component from being shared by more than one n-tuple.

[3] The comma is unconventional but is used here optionally for extra clarity.

(Equivalently, it is a network of N RAMs, each containing a C-dimensional bit vector.) The memory content $m_{ci\alpha}$ at address α of the $i^{\underline{\text{th}}}$ node allocated to class c is set to

$$m_{ci\alpha} = \Theta_\theta \left(\sum_{\boldsymbol{v} \in \mathcal{D}_c} \delta_{\alpha,\alpha_i(\boldsymbol{v})} \right). \tag{4}$$

Note, that symbol α denotes any address value in range $[0, 2^n - 1]$ and $\alpha_i(\boldsymbol{u})$ refers to one particular address value gererated by the input pattern $\boldsymbol{u}$.

In the usual $\theta = 1$ case, the 1-bit content of $m_{ci\alpha}$ is set if any pattern of $\mathcal{D}_c$ has feature α_i and unset otherwise. Recognition is accomplished by summing the contents of the nodes of each class at the addresses given by the features of the unclassified pattern. That is, pattern $\boldsymbol{u}$ is assigned to class

$$\underset{c}{\text{argmax}} \left(\sum_{i=1}^{N} m_{ci\alpha_i(\boldsymbol{u})} \right). \tag{5}$$

To generalise the n-tuple classifier to a function interpolator, a vector $\boldsymbol{m}_{i\alpha}$ of the appropriate numeric type for the range of the function is stored at each address α in each node i. Training is accomplished by assigning these vectors to the averages

$$\boldsymbol{m}_{i\alpha} = \sum_{\boldsymbol{v} \in \mathcal{D}_c} \boldsymbol{Y}(\boldsymbol{v}) \delta_{\alpha,\alpha_i(\boldsymbol{v})} / \sum_{\boldsymbol{v} \in \mathcal{D}_c} \delta_{\alpha,\alpha_i(\boldsymbol{v})}, \tag{6}$$

where $\boldsymbol{Y}(\boldsymbol{v})$ is the desired output for input bit string $\boldsymbol{v}$. The network output, or response to an arbitrary input pattern $\boldsymbol{u}$ is

$$\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{m}_{i\alpha_i(\boldsymbol{u})}. \tag{7}$$

The $\theta = \infty$ version of the classifier (4), (5) is recovered when $\boldsymbol{Y}(\boldsymbol{v})$ is the indicator function for the class $c(\boldsymbol{v})$ of $\boldsymbol{v}$: $\boldsymbol{Y}_c(\boldsymbol{v}) = \delta_{c,c(\boldsymbol{v})}$. Experimental experience with the $\theta = \infty$ version of the classifier is disappointing (Bledsoe & Bisson, 1962), possibly for the reasons given in section 3.3, and this may bode ill for the interpolator, which is little studied. However, the interpolator is at least of theoretical interest for its connection with the Kanerva model explained in section 3.2.1.

2.2 Practical experience

Simple theoretical considerations and practical experience provide fairly strong guidance for setting the architectural parameters, n, N, and θ. To begin with, the fact that the network response to an arbitrary pattern is essentially an average over the n-tuples (5) means that the results should become increasingly consistent with increasing N. Because the n-tuple method can process thousands of patterns within seconds, there is little need for any data analysis method other than explicit measurement of the variation of performance as the input mapping is re-randomised a few times for a given N. N is increased if the variation is unacceptably high. Practical experience is that values of 100 to 1000 usually turn out to be adequate. It is commonly observed that the network's output for the winning class exceeds that of the second runner up by an uncomfortably small margin, such as 3 n-tuples out of 1000, but that the correct class nevertheless wins consistently. Perhaps most n-tuples give a constant response to most patterns, so effectively only a fraction of those in the network are contributing to the decisions. This suggests that many n-tuples could be trimmed from the network, but such variations on the method are vulnerable to overtraining and complicate the theory. The phenomenon deserves further study.

Practical experience tends to favour small values of the threshold θ, particularly $\theta = 1$. A possible rationale for this is given in section 3.3. Many considerations apply to the choice of n-tuple size n. Experimentally it usually turns out that bigger is better, up to an impractically large size (Rohwer & Lamb, 1993), which requires an unreasonable amount of training data, but $n = 8$ is usually enough, and $n = 3$ is sometimes adequate. This can be explained qualitatively by observing that information about the correlations among up to n bits are available to the classifier. It never hurts to take account of higher-order correlations, but it is plausible that $8^{\underline{\text{th}}}$ order correlations contain all that is needed for most binary data

sets. Another intuition is that the training process should write to neither too small nor too great a proportion of the 2^n addresses at each node. If n is too large, the sub-patterns occuring in the training data will be unlikely to recur in the test data, whereas if n is too small, the memory can *saturate*, in which case $m_{c\alpha_i} = \theta$ for most memory locations, so most discriminative power is lost (Ullmann, 1969), (Tarling & Rohwer, 1993). These issues are further complicated if the class priors are highly skewed, so that one class has far more training data than another. Although a precise theory is not available, there are strong enough theoretical tools to gain considerable insight, as is demonstrated in the discussion of the experimental results.

3 Theoretical Status of the n-tuple method

The n-tuple classifier is a memory-based method. Such methods differ from optimisation-based methods, such as Back Propagation of error through multi-layer perceptrons, in two important ways. Firstly, "hidden" representations (or "features") are selected randomly, and secondly, training is a simple one-shot memorisation task involving these features. These differences give memory-based methods an awesome advantage in training speed. Radial Basis Functions obtain part of this speed advantage by selecting features randomly (Broomhead & Lowe, 1988), and multi-layer perceptrons can often be trained faster with little or no loss of performance by using fixed random weights into the hidden layers (Gallant & Smith, 1987; Sutton & Whitehead, 1993). However, this does not give the speed and simplicity that training by mere memorisation provides.

In spite of many useful advances (Austin, 1994; Aleksander & Stonham, sh1979a; Flanagan *et al.*, 1992; Bledsoe & Bisson, 1962; Ullmann & Kidd, 1969), there is no theory of n-tuple networks of the standard of the sophisticated statistical techniques available with

optimisation-based methods (MacKay, 1992). It is not particularly difficult to design new training algorithms for the n-tuple architecture in order to make these statistical methods applicable (Tattersall *et al.*, 1991; Luttrell, 1992; Rohwer, 1995), but this approach sidesteps the interesting questions instead of answering them. These modified methods reduce the speed and simplicity advantages as well.

The main tools for understanding the standard n-tuple network are reviewed here, and used to place the n-tuple net into context with similar methods. These tools are used again to explain the experimental results.

3.1 Approaches based on Hamming Distance

The most productive theoretical concept for understanding the n-tuple method has been "tuple distance" and its nonlinear statistical relationship with Hamming distance.

3.1.1 Tuple distance and Hamming distance

The main theoretical results on the n-tuple method provide a relationship between a "*tuple distance*" relevant to the network's generalisation properties, and the Hamming distance between training and test patterns. The tuple distance $\rho(\boldsymbol{u},\boldsymbol{v})$ between patterns $\boldsymbol{u}$ and $\boldsymbol{v}$ is the number of tuples (of a given input mapping) on which the patterns disagree:

$$\rho(\boldsymbol{u},\boldsymbol{v}) = N - \sum_{i=1}^{N} \delta_{\alpha_i(\boldsymbol{u}),\alpha_i(\boldsymbol{v})}. \tag{8}$$

The number on which they agree, $N-\rho(\boldsymbol{u},\boldsymbol{v})$, will be called the "tuple score." An elementary argument based on the random selection of the n-tuple inputs from the L bits available shows that patterns $\boldsymbol{v}$ which lie a fixed Hamming distance $H(\boldsymbol{u},\boldsymbol{v})$ from any one pattern $\boldsymbol{u}$ are distributed binomially in tuple distance:

$$P(\rho|H) = \frac{N!}{\rho!(N-\rho)!}\left(1-\left(1-\frac{H}{L}\right)^{n}\right)^{\rho}\left(1-\frac{H}{L}\right)^{n(N-\rho)}. \tag{9}$$

More complicated expressions are available for more constrained n-tuple sampling procedures (Tattersall & Johnson, 1984). Distribution (9) gives an expectation value for ρ of

$$\rho(H) = \langle \rho | H \rangle = N \left(1 - \left(1 - \frac{H}{L} \right)^n \right), \tag{10}$$

and indicates that ρ typically strays from this value by the standard deviation

$$\delta\rho(H) = \left[N \left(1 - \frac{H}{L} \right)^n \left(1 - \left(1 - \frac{H}{L} \right)^n \right) \right]^{\frac{1}{2}}. \tag{11}$$

This is illustrated in figure 1. If the patterns are nearby ($H \ll L$), then a convenient approximation is

$$\rho(H) \approx N \left(1 - e^{-n\frac{H}{L}} \right). \tag{12}$$

The n-tuple sampling variations then make $\rho(H)$ uncertain by about $\sqrt{\rho(H)}$.

It is clear from (12) that proximity in Hamming distance plays a role in the generalisation behaviour of n-tuple networks. Consider a network trained on just one example $\boldsymbol{v}$ of class c, and tested on a pattern $\boldsymbol{u}$ Hamming distance H from $\boldsymbol{v}$. Classifications are based on the network response $\sum_{i=1}^{N} m_{ci\alpha_i(\boldsymbol{u})}$ to pattern $\boldsymbol{u}$, which will be about $Ne^{-n\frac{H(\boldsymbol{u},\boldsymbol{v})}{L}}$. Hence one could say that the network generalises from training pattern $\boldsymbol{v}$ to all patterns within a Hamming distance of about L/n of $\boldsymbol{v}$. Figure 1 shows a "clean" case in which the training patterns within L/n of a test pattern are predominately of the correct class. In our experience it is far more usual for many incorrect patterns to populate this region as well, but the classifier seems to work anyway. It would appear that the training patterns nearest a test pattern tend to match on tuples ("overlap") no more often than patterns of any other class.

3.1.2 Training Data overlap on tuples

A network trained on a set of patterns $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_T\}$ could respond to pattern $\boldsymbol{u}$ by any amount between $N \min(1, \max_a e^{-n\frac{H(\boldsymbol{u},\boldsymbol{v}_a)}{L}})$ and $N \min(1, \sum_a e^{-n\frac{H(\boldsymbol{u},\boldsymbol{v}_a)}{L}})$, depending on the correlations between the training patterns, as manifest in "overlap effects". Unfortunately, this circumstance limits the usefulness of tuple distance for explaining the standard n-tuple method. Because of a combinatorial explosion, there is no feasible method of measuring tuple correlations. Similar problems crippling an attempt of full formal analysis of the method (Stonham, 1977) for datasets of arbitrary size have been reported. However, some insight into the mechanism of the RAMnet can be gained by analysis of the experimental data.

Overlap effects are displayed in figure 2. Figures 2a and 2b show the network output for a test pattern as training patterns are added in Hamming distance order from the test pattern. For figure 2a, the threshold is $\theta = 1$, and for figure 2b it is $\theta = \infty$, effectively ignoring overlap in that the tuple scores are added for all the training patterns. Figure 2b shows that distant patterns of the incorrect class match the test pattern on many n-tuples, but figure 2a shows that most of these subpatterns had already turned up in closer training patterns. Figure 2c shows the number of new RAM locations accessed as training patterns are accumulated, regardless of whether these are accessed by the test pattern. One class is disadvantaged by a smaller prior probability, and correspondingly fewer training samples, but it has the advantage that it populates new RAM locations more rapidly. Whereas the more probable class is showing signs of levelling off in this respect, the less probable class is not, so it seems likely that if more data were available (in the same proportions), then the test pattern would be more likely to be classified correctly with the less probable class. This scenario was frequently observed in datasets with skewed priors, and motivates the

hypothesis that errors would have been reduced if more data were available.

3.2 Related methods

Tuple distance can be used to compare the n-tuple network to Kanerva's sparse distributed memory, the n-tuple regression network or single-layer lookup perceptron, and the familiar radial basis functions network. The n-tuple network can also be interpreted as a crude likelihood estimator.

3.2.1 The Kanerva Model

The n-tuple network is related to Kanerva's Sparse Distributed Memory model (Kanerva, 1988; Songcan & Jun, 1992), which has also been developed theoretically using an "overlap" measure similar to (12). Although intended mainly as an associative memory, it is easily generalised for classification problems or function interpolation problems. The interpolation version is presented here.

Instead of n-tuples, a set of N bit strings are randomly selected from a uniform distribution. These are used as *centres* $\boldsymbol{\xi}_i$ of hard-sphere radial basis functions ϕ_i of binary vectors $\boldsymbol{v}$

$$\phi_i(\boldsymbol{v}; r) = \begin{cases} 1 & H(\boldsymbol{\xi}_i, \boldsymbol{v}) \leq r) \\ 0 & H(\boldsymbol{\xi}_i, \boldsymbol{v}) > r) \end{cases} \tag{13}$$

with a somewhat carefully chosen radius r. Memory space for a vector in the range of the function to be approximated is associated with each centre. The memory at centre i is set to

$$\boldsymbol{m}_i^{(\mathrm{K})} = \frac{\sum_{\boldsymbol{v} \in \mathcal{D}} \boldsymbol{Y}(\boldsymbol{v}) \phi_i(\boldsymbol{v}; r)}{\sum_{\boldsymbol{v} \in \mathcal{D}} \phi_i(\boldsymbol{v}; r)} \tag{14}$$

during training. Here $\boldsymbol{Y}(\boldsymbol{v})$ is the desired output for input pattern $\boldsymbol{v}$. The output $\boldsymbol{Y}(\boldsymbol{v})$ and memory $\boldsymbol{m}_i^{(\mathrm{K})}$ can belong to any space in which a weighted average can be defined.

For classification problems, $\boldsymbol{Y}(\boldsymbol{v})$ is an indicator function, and for an associative memory, $\boldsymbol{Y}(\boldsymbol{v}) = \boldsymbol{v}$.

The network response to test pattern $\boldsymbol{u}$ is

$$\boldsymbol{y}^{(\mathrm{K})}(\boldsymbol{u}) = \frac{\sum_i \boldsymbol{m}_i^{(\mathrm{K})} \phi_i(\boldsymbol{u}; r)}{\sum_i \phi_i(\boldsymbol{u}; r)}, \tag{15}$$

which is

$$\boldsymbol{y}^{(\mathrm{K})}(\boldsymbol{u}) = \frac{\sum_{\boldsymbol{v} \in \mathcal{D}} \boldsymbol{Y}(\boldsymbol{v}) \sum_i \phi_i(\boldsymbol{v}; r) \phi_i(\boldsymbol{u}; r)}{\sum_{\boldsymbol{v} \in \mathcal{D}} \sum_i \phi_i(\boldsymbol{v}; r) \phi_i(\boldsymbol{u}; r)} \tag{16}$$

in terms of the training data. (A further thresholding operation is required if a the output is to be a bit string.) Each training pattern $\boldsymbol{v}$ contributes its desired output $Y_c(\boldsymbol{v})$ to an average weighted by

$$N - \rho_r^{(\mathrm{K})}(\boldsymbol{v}, \boldsymbol{u}) \stackrel{\mathrm{def}}{=} \sum_i \phi_i(\boldsymbol{v}; r) \phi_i(\boldsymbol{u}; r), \tag{17}$$

the number of centres within Hamming distance r of both the training pattern and the test pattern. Evidently, $N - \rho_r^{(\mathrm{K})}(\boldsymbol{v}, \boldsymbol{u})$ plays a role similar to the tuple score, with centres within distance r of both patterns being counted instead of n-tuples. Due to the random placement of centres, the expectation of $N - \rho_r^{(\mathrm{K})}(\boldsymbol{v}, \boldsymbol{u})$ is also a function $\rho_r^{(\mathrm{K})}(H(\boldsymbol{u}, \boldsymbol{v}))$ of the Hamming distance $H(\boldsymbol{v}, \boldsymbol{u})$, although it is more complicated than (10) or (12). The exact form is a sum of products of binomial coefficients which can be approximated by

$$\rho_r^{(\mathrm{K})}(H) \approx N \left(1 - \int_{H/L}^{1} \frac{dx}{2\pi \sqrt{x(1-x)}} e^{-\frac{(r - L/2)^2}{L/2} \frac{1}{(1-x)}} \right) \tag{18}$$

for $0 \ll H \ll L$.

To carry the comparison further, observe that the interpolative n-tuple network (6), (7) can be regarded as a special case of the Kanerva network (14), (15), if the Hamming distance in (14) and (15) is replaced with a Hamming distance restricted to a tuple. Specifically, a Kanerva centre $\boldsymbol{\xi}_{i\alpha}$ can be associated with each memory location α at each n-tuple i by

defining all bits of $\boldsymbol{\xi}_{i\alpha}$ arbitrarily except for those involved in the $i^{\underline{\text{th}}}$ input mapping. These bits must form subpattern α: $\sum_{j=0}^{n-1} \xi_{\eta_i(j),\alpha} 2^j = \alpha$. The Hamming distance is replaced by $\sum_{j=0}^{n-1} (1 - \delta_{\boldsymbol{\xi}_{\eta_i(j)}, \boldsymbol{u}_{\eta_i(j)}})$ and $r = 0$.

3.2.2 The n-tuple regression network

The fact that the n-tuple network response to a test pattern is dominated by the (Hamming) nearest training patterns suggests that it might be formally related to a nearest neighbour method or a local basis function method. A modified method makes the most of this resemblance. Called the *n-tuple regression network* (Allinson & Kołcz, 1994b), or the single-layer lookup perceptron (Tattersall *et al.*, 1991), it treats the interpolative n-tuple network response (7) as a weighted sum of basis functions:

$$y_c = \sum_{i=1}^{N} m_{ci\alpha_i(\boldsymbol{u})} = \sum_{i=1}^{N} \sum_{\alpha=0}^{2^n-1} m_{ci\alpha} \delta_{\alpha,\alpha_i(\boldsymbol{u})}. \tag{19}$$

From the latter form, the network can be regarded as a linear transformation applied to the outputs of the rather unusual basis functions $\delta_{\alpha,\alpha_i(\boldsymbol{u})}$, one for each combination of i and α. Training is easily accomplished by a least mean squares (LMS) method.

Expression (19) can be related to a basis function expansion in the the exponentials appearing in (12),

$$y_c = \sum_a w_{ca} e^{-n\frac{H(\boldsymbol{u},\boldsymbol{v}_a)}{L}} \tag{20}$$

provided that patterns $\boldsymbol{v}_a$ and weights w_{ca} can be found such that

$$m_{ci\alpha} = \sum_a w_{ca} \delta_{\alpha,\alpha_i(\boldsymbol{v}_a)}. \tag{21}$$

(Plugging (21) into (19) and using (8) gives (20) within approximation (12).)

One way to arrange this is to choose all the patterns $\boldsymbol{v}_a$ so they are separated from each other by tuple distance N; *ie.*, none of the patterns $\boldsymbol{v}_a$ match each other in any n-tuple.

(This is possible if and only if the number of these patterns is no more than 2^n.) In this situation there is at most one a for any given n-tuple i and address α_i such that $\alpha_i(\boldsymbol{v}_a) = \alpha$, which may be called $a(i, \alpha)$ when it exists. Then the choice

$$m_{c\alpha_i} = \begin{cases} w_{ca(i,\alpha)} & a(i,\alpha)\text{defined} \\ 0 & a(i,\alpha)\text{not defined} \end{cases} \tag{22}$$

satisfies (21). Such a network acts like the radial basis function network (20) with a (Hamming[4]) spherically symmetric local basis function of radius roughly L/n centred on each pattern $\boldsymbol{v}_a$, even though the patterns $\boldsymbol{v}_a$ do not appear in the implementation (19), and to the benefit of computation speed, no distance calculations $H(\boldsymbol{u}, \boldsymbol{v}_a)$ are ever performed.

If the training patterns meet the separation condition, they can be identified with the basis function centres $\boldsymbol{v}_a$. To see this, observe that the assumed one-to-one correspondence between pattern a and address $\alpha_i(\boldsymbol{v}_a)$ in the $i^{\underline{\text{th}}}$ n-tuple memory implies that the set of memory locations corresponding to one pattern is disjoint from the set addressed by any other. Therefore the LMS optimisation problem involving (19) will be symmetric with respect to the N parameters $m_{ci\alpha_i(\boldsymbol{v}_a)}$ for each a and c. Hence these parameters will all be equal in the solution, and one can meaningfully define $m_{ca} = m_{ci\alpha_i(\boldsymbol{v}_a)}$. The network response to a test pattern (19) can then be re-written by replacing the sum over addresses with a sum over training patterns:

$$y_c = \sum_{i=1}^{N} \sum_{a} m_{ci\alpha_i(\boldsymbol{v}_a)} \delta_{\alpha_i(\boldsymbol{v}_a),\alpha_i(\boldsymbol{u})}. \tag{23}$$

Then it is clear from (8) and (20) that one can identify $m_{ca} = w_{ca}$.

The interpretation of an n-tuple regression network as an effective radial basis function network with a function centre on each training pattern requires the patterns $\boldsymbol{v}_a$ to be tuple-

[4]The method can be modified so that the effective basis functions have more nearly Euclidean-spherical receptive fields (Kolcz & Allinson, 1994).

separated by N. This condition would be valid at least to a good approximation if their Hamming separation were large compared to L/n. The analysis of our experiments typified by figures 1 and 7 does not suggest that this is likely to happen, and even in principle it may not be easy to arrange this *and* ensure good coverage of the pattern space by the local effective basis functions. For this, the data needs to just happen to be arranged so that each training pattern of a class is about L/n bits away from it's nearest neighbour. To do as well as possible on both conditions, all other neighbours should be much further away, but the triangle inequality requires the next nearest neighbour to be within $2L/n$ bits. If the data just happened to be suitably arranged for this interpretation, then a suitable choice of n-tuple size n could be stated in terms of the nearest neighbour distance d_{NN} as $n = L/d_{\text{NN}}$.

Even if the training data is not distributed so that a basis function interpretation is possible with the training points serving as function centres, it may still be possible to obtain a basis function interpretation by finding a solution, at least approximately, to (21). In any case, there is no reason to suppose that the method will fail when the interpretations do.

3.3 The n-tuple classifier as a crude likelihood estimator

An alternative to using tuple distance to formulate a theory of n-tuple networks is to view them as probability estimators. With $\theta = \infty$, expression (4) for the memory content $m_{ci\alpha}$ can be interpreted as an estimate of the probability $P_i(\alpha|c)$ (up to a normalisation factor) that a data point from a given class c will have subpattern α in n-tuple i. Assuming these distributions for different n-tuples to be independent of each other, the joint distribution over all the sub-patterns taken together is

$$P(\boldsymbol{\alpha}|c) = \prod_i P_i(\alpha|c). \tag{24}$$

This can be used for maximum likelihood classification, or converted to posterior class probabilities using Bayes' rule with class priors $P(c)$, if they are available. The independence assumption lacks plausibility, because it would be remarkable for the correlations required to make the classes distinguishable not to be reflected in correlations between the n-tuples, but nevertheless, good results have been reported with this formulation, which has been re-invented from time to time (Bledsoe & Bisson, 1962; Sixsmith *et al.*, 1990; Badr, 1993). Aside from its implausibility, its main practical problem is that factors of zero appear if a naive estimate of $P_i(\alpha|c)$ is used for subpatterns which never appear in the training data. In practice these are replaced with a small *ad hoc* positive constant, say ϵ, leaving scope for more principled approaches to estimating these probabilities.

The n-tuple method with finite threshold can be seen as a scaled and translated approximation to the logarithm of (24). Specifically, if test pattern $\boldsymbol{u}$ has sub-patterns $\boldsymbol{\alpha}(\boldsymbol{u})$, and the T_c training patterns from class c supply the tallies $T_{ci\alpha} = \sum_{\boldsymbol{v}\in\mathcal{D}_c} \delta_{\alpha,\alpha_i(\boldsymbol{v})}$ used to estimate $P_i(\alpha|c)$ by $T_{ci\alpha}/T_c$, then for suitable choices of ϵ and θ, the network responses in (5) will approximately satisfy

$$\begin{aligned} N + \frac{\log\left(T_c^N P(\boldsymbol{\alpha}(\boldsymbol{u})|c)\right)}{-\log\epsilon} &= \sum_{i=1}^{N}\left(1 + \frac{\log T_c + \log P_i(\alpha(\boldsymbol{u})|c)}{-\log\epsilon}\right) \\ &= \sum_{i=1}^{N}\left(1 + \frac{\log T_{c\alpha_i}}{-\log\epsilon}\right) \approx \sum_{i=1}^{N}\Theta_\theta(T_{c\alpha_i}) \end{aligned} \tag{25}$$

as illustrated by figure 3. For integer tallies, the approximation becomes arbitrarily accurate for $\theta = 1$ as $\epsilon \to 0$. Hence the standard n-tuple method could be justified this way if the independence assumption were acceptable and the absence of sub-patterns in the training data could be taken as strong evidence that the corresponding probabilities are tiny. Essentially, the method counts the number of factors of ϵ in (24).

4 Preprocessing of scalar attributes

A RAMnet classifies bit strings, but the attributes of the patterns in the StatLog data sets are mostly real numbers or integers. Given that generalisation from numerical attributes should be related to arithmetic differences, and generalisation in RAMnets is related to Hamming distances, it is important to transform numbers into bit strings in such a way that numerical proximity is transformed into Hamming proximity. A memory-efficient method tailored to the generalised Hamming distance underlying RAMnet generalisation has been provided by Allinson (Allinson & Kołcz, 1993), (Allinson & Kołcz, 1994a), using a combination of CMAC and Gray coding techniques. The prescription for encoding integer x is to concatenate K bit strings, the $j\underline{\text{th}}$ of which (counting from 1) is $\frac{x+j-1}{K}$, rounded down and expressed as a Gray code. The Gray code of an integer i can be obtained as the bitwise exclusive-or of i (expressed as an ordinary base 2 number) with $i/2$ (rounded down). This provides a representation in aK bits of the integers between 0 and $(2^a-1)K$ inclusive, such that if integers x and y differ arithmetically by K or less, their codes differ by Hamming distance $|x-y|$, and if their arithmetic distance is K or more, their corresponding Hamming distance is at least K. This is illustrated in figure 4, and more comprehensive illustrations are given by (Allinson & Kołcz, 1993).

Because tuple score decays exponentially with Hamming distance 12, there should be relatively little ill effect if a training pattern further than L/n bits away from a test pattern is replaced by another training pattern further than L/n bits away (although figures 1 and 7 indicate that L/n is a *very* approximate estimate). Thus if there are A scalar attributes, one can expect the non-linearity of the CMAC/Gray mapping to do little harm if $K > \frac{L/A}{n}$.

There are aK bits per attribute, so this condition is

$$a < n. \tag{26}$$

Scalar differences up to $\pm K$ fall within the linear region of the mapping. This represents a fraction $\frac{2K}{(2^a-1)K}$ or about 2^{1-a} of the largest separation allowed. With $a < n$, the "generalisation Hamming distance" $\frac{L/A}{n} = \frac{aK}{n}$ corresponds to a scalar separation of $\pm\frac{aK}{n}$, which is the fraction $\frac{2a}{n(2^a-1)} \approx \frac{a}{n}2^{1-a}$ (for $a > 1$) of the largest possible scalar separation.

For $a = 1$, the mapping becomes the "thermometer code", in which integer x is mapped to a bit string with the last x bits set and the remaining $K - x/K$ unset. If K is adjusted to preserve the input interval, then larger a values give shorter codes, which should be similarly effective as long as $a < n$ and scalar attributes separated by more than fraction 2^{1-a} of their dynamic range can be regarded as dissimilar as far as generalisation is concerned.

Figure 5 shows the test set classification accuracy as a function of n for a 100-bit thermometer code, a 48-bit ($a = 3$, $K = 16$) code and a 40-bit ($a = 5$, $K = 8$) code for one of the datasets studied. For small values of n, the longer codes perform better, presumably because they are linear over a larger fraction of the dynamic range. But with increasing n, the "generalisation distance" shrinks, so more and more of the linear region of the longer codes is ignored by the RAMnet, eliminating their advantage. Meanwhile, accuracy improves with n, eventually levelling out. This is probably due to making higher-order correlations available, as well as reducing saturation effects.

5 The experiments

Extensive experimental trials were conducted in order to benchmark and study the n-tuple algorithm.

5.1 Selection and pre-processing of StatLog data sets

The European Community ESPRIT project 5170, the StatLog project, was designed to carry out comparative testing and evaluation of classification algorithms on large scale applications. About 20 data sets were used to estimate the performance of 23 procedures. These are described in detail in (Michie *et al.*, 1994). Each of the larger data sets (with many more than 1000 samples) were randomly split into training and testing partitions. Different methodologies (cross-validation and bootstrap) were applied to the smaller data sets. This study used the large data sets, which are summarised in table 2. There are 11 of these.

5.2 Experimental details

The CMAC/Gray parameters used were $K = 8$ and $a = 5$, giving 40-bit representations of the integers in $[0, 248]$. All scalar attributes were linearly rescaled and rounded to obtain integers in this interval. In the Letter data set (See Table 2), where the attributes can take on only 16 values, it would be more reasonable to use a one-out-of-N encoding with strings of 16 bits, but the CMAC/Gray procedure was used anyway for the convenience of uniformity.

The threshold θ was set to 1 in all the experiments reported here, the n-tuple size n was set to 8, and N was set to 1000 n-tuples. These values for n and N yield the best classification results and have been determined experimentaly as discussed in section 2.2. The results reported are averages over 10 different random input mappings η. Using $n = 6$ gave similar results.

5.3 Time and Memory requirements

Computation time requirements were insignificant in these experiments, which were carried out with a C++ program on a SUN Sparc workstation. For example, an 8-tuple network can be trained on the 2000 57-attribute training patterns of the BelgianII data set in about 49 seconds. Sixteen of these seconds are needed just to read in the data; another 4 to do the CMAC/Gray conversion of the floating point attributes; and the final 29 to train the RAMnet itself. Testing the same 2000 patterns takes slightly longer, 37 seconds instead of 29, because a loop over classes is needed within the loop over n-tuples. Detailed timing statistics are not published for the algorithms used in the StatLog project, but it is clear that popular neural network algorithms such as Back Propagation and even the relatively fast Radial Basis Functions are slow by comparison. The algorithm is highly parallelisable, so if it were important for the RAMnet to be even faster, special purpose parallel hardware could be designed or purchased (Aleksander *et al.*, 1984). It would be feasible for a biological system to implement a highly parallel but otherwise trivial calculation along these lines.

The storage requirements were moderate in most cases. In the most extreme case (Shuttle) 128kB of RAM per class was needed.

5.4 Results

The classification results for each algorithm attempted with each data set are presented in figure 6. Table 1 gives a brief description of each algorithm with the symbol used to represent it in the figure. The classification error rates increase from left to right, and are scaled separately for each data set, so that they equal 1 at the error rate of the trivial method of always guessing the class with the highest prior probability, ignoring the input pattern.

As remarked in section 5.2, the results plotted for the n-tuple recognition algorithm are averages over 10 randomly selected input mappings. If the corresponding standard deviations were plotted as error bars in figure 6, they would be obscured by the dots representing the means.

Algorithm codes appear in Table 1. Classification error rates increase from left to right, and are scaled separately for each data set, so that they equal 1 at the error rate of the trivial method of always guessing the class with the highest prior probability, ignoring the input pattern. The arrows indicate the few cases in which performance was worse than this.

6 Analysis of Results

The n-tuple method delivered competitive accuracy on 6 of the data sets tested (Shuttle, Letter, Tsetse, BelgianI, Chromo, SatIm), performed modestly on 1 (DNA) and failed entirely on the other 4 (Belgian II, Cut50, Cut20, Technical). Further experimental and theoretical analysis was carried out to explain the failures.

The available tuple-distance theory is not amenable to treating overlap effects, even though figures 2, 1 and 7 (and many like them) show these to be important. Therefore we generated and inspected many detailed Hamming distance vs. tuple score plots like figure 1 in order to develop a qualitative impression of the validity of this theory when there are many training patterns.

Ignoring overlap effects, a test pattern should be assigned to the class of most of the training patterns that lie nearer to it, in Hamming distance, than about L/n. A glance at figure 7 shows that the distribution of tuple distances bears no systematic relationship to L/n, but nevertheless it was found that the closest patterns did tend to determine the decision, at least when the class priors were roughly equal. Figures 7a and 7b show

examples of this, which was by far the most common situation encountered. Figure 7c shows an error due to overlap effects, abetted by a highly skewed prior. The 4 troublesome datasets had highly skewed priors and predominately showed this pattern, although the very easy Shuttle dataset also had highly skewed priors. Figure 7d shows a relatively rare situation in which overlap effects rescued a pattern which would have been misclassified, judging by its Hamming-near neighbours. It would appear that although overlap effects are quantitatively important, they tend not to alter the conclusion that the near neighbours determine the decision, at least when the priors are relatively uniform.

Given that Hamming neighbours tend to determine the classification outcome, it seems sensible to suspect that test patterns in the 4 problematic data sets have a shortage of good neighbours. It turns out that they simply don't have enough neighbours at all, within the distance scales relevant to RAMnet generalisation. To generalise properly, a test pattern must have at least 1 training pattern within a Hamming distance of about L/n. Distributed evenly over A CMAC/Gray-mapped scalar attributes, this is a scalar difference of about $\frac{a}{n}2^{1-a}$, with the attributes scaled to lie between 0 and 1, as explained in section 4. Therefore each training pattern can provide information about any test pattern which falls within a hypercube of volume roughly $(\frac{a}{n}2^{1-a})^A$. The number of such cubes required to cover the region of attribute space where test data is likely to appear can be crudely estimated by approximating this region as a hyper-rectangle with edge lengths given by the eigenvalues of the sample covariance matrix of the training data. Any eigenvalues smaller than $\frac{a}{n}2^{1-a}$ should be rounded up to this value, because the covering cubes must be at least this thick. The number of "generalisation hypercubes" required to cover the data region is therefore roughly $\prod_{i=1}^{A} \max(1, \lambda_i \frac{n}{a} 2^{a-1})$ for $1 < a \leq n$, where the λ_i are the eigenvalues. Figure 8 shows this lower bound on the number of training samples required, for each dataset

studied, taking $a = 5$ and $n = 8$ as in the experiments.

Aside from Technical and DNA, the problematic datasets stand out as several orders of magnitude more deficient in training data than the others, some of which are mildly deficient according to this crude estimate. DNA is special in that its Boolean attributes were treated as integers, so it's data distribution will be highly non-Gaussian and therefore poorly described by the covariance matrix. The Technical data set turned out to be coverable by just 1 hypercube, according to this estimate. Presumably then, each of its patterns looks the same to the RAMnet, and this accounts for its failure. Perhaps a non-linear rescaling of its attributes, such as histogram equalisation, would help. This possibility remains to be pursued.

It is not possible to address the data deficiencies by supplying more data, especially when several orders of magnitude more samples are needed, but it is possible to tweak the RAMnet parameters to enlarge the "generalisation cubes". However, there is less room to maneuver than one would like. To enlarge the cubes, n must be decreased, but this risks degradation of performance due to loss of high-order correlation information, as indicated in figure 5. Decreasing n also requires decreasing a, if the constraint $a \leq n$ is to be respected, keeping L/n within the linear region of the CMAC/Gray mapping. Low a values give less memory-efficient representations of scalars, at any given resolution. Systematic experiments varying the parameters did not produce significant improvements on the 4 problematic data sets (or the others). A more far-reaching improvement in the algorithm is required.

7 Conclusions

Extensive experimental trials, on a scale uncommon for *any* algorithm, were carried out with the n-tuple classifier. The fact that this was possible at all testifies to the method's

speed, which derives from its simple principle of learning by 1-shot memorisation of random features. In 6 of the 11 datasets tested, this speed and simplicity can be enjoyed without sacrificing classification accuracy relative to 23 other slower methods, including the most popular neural network methods. Clearly, the n-tuple algorithm should be considered by every neural network researcher before a more sophisticated and therefore slower method is applied.

The theoretical tools available for the n-tuple method were reviewed and used to place it into context with other algorithms. These tools were also used to explain what went wrong on the 4 data sets which gave poor results. One problem is that skewed class priors tend to be problematic in a way that is difficult to quantify using current theories of "overlap effects". A more fundamental problem is that one parameter, the n-tuple size n, controls both the distance-scales associated with generalisation behaviour, and the complexity of the random features used to discriminate classes. For some data sets it is not possible to find one setting suitable for both of these considerations.

In spite of its imperfections, the n-tuple method demonstrates that its underlying principle, learning by memorisation of random features, is a powerful one. It should be rewarding to develop the theory further, especially by inventing practical approximations to describe overlap effects, and to invent improved methods which incorporate the underlying principle in a more flexible way.

8 Acknowledgements

The authors are grateful to Louis Wehenkel of Universite de Liege for useful correspondence and permission to report results on the BelgianI and BelgianII data sets, Trevor Booth of the Australian CSIRO Division of Forestry for permission to report results on the Tsetse

data set, and Reza Nakhaeizadeh of Daimler-Benz, Ulm, Germany for permission to report on the Technical, Cut20 and Cut50 data sets.

References

Aleksander, I., & Stonham, T. J. (1979a). Guide to pattern recognition using random-access memories. *Computers and Digital Techniques*, **2**, 29–40.

Aleksander, I., & Stonham, T.J. (1979b). Guide to pattern recognition using random–access memories. *IEE Proceedings on Computers and Digital Techniques (part E)*, **2**(1), 29–40.

Aleksander, I., Thomas, W. V., & Bowden, P. A. (1984). WISARD: A Radical Step Forward in Image Recognition. *Sensor Review*, **4**, 120–124.

Alexander, I., & Morton, H. (1995). *An Introduction to Neural Computing.* Thomson Publishing. Chap. 5. The secrets of the WISARD.

Allinson, N.M., & Kołcz, A. (1993). Enhanced N-tuple approximators. *Weightless Neural Network Workshop*, 38–45.

Allinson, N.M., & Kołcz, A. (1994a). Application of the CMAC input encoding scheme in the n-tuple approximation network. *IEE Proceedings on Comput. Digit. Tech.*, **141**(3), 177–183.

Allinson, N.M., & Kołcz, A. (1994b). The theory and practice of N-tuple neural networks. *In:* Taylor, J.G. (ed), *Adaptive Computing and Information Processing:Neural Networks.* Unicom Publishing.

Austin, J. (1994). A Review of RAM based Neural Networks. *Pages 58–66 of: Proc. of the Fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems.* Turin: IEEE Computer Society Press.

Badr, Ardeshir. (1993). N-Tuple Classifier for ECG Signals. *Pages 29 – 32 of:* Allinson, N. (ed), *Proceedings of the Weightless Neural Network Workshop '93, Computing with Logical Neurons.*

Bledsoe, W. W., & Bisson, C. L. (1962). Improved Memory Matrices for the n-Tuple Pattern Recognition Method. *IEEE Trans. Electronic Computers*, **11**, 414–415.

Bledsoe, W. W., & Browning, I. (1959). Pattern recognition and reading by machine. *Pages 232–255 of: Proceedings of the Eastern Joint Computer Conference.*

Broomhead, D. S., & Lowe, David. (1988). Multi-variable functional interpolation and adaptive networks. *Complex Systems*, **2**, 321–355.

Flanagan, C., Rahman, M. A., & McQuade, E. (1992). A Model for the Behaviour of N-Tuple RAM Classifiers in Noise. *Journal of Intelligent Systems*, **2**, 187–224.

Gallant, S., & Smith, D. (1987). Random Cells: an idea whose time has come and gone... and come again? *Pages II–671–II–678 of:* Caudill, & Butler (eds), *IEEE International Conference on Neural Networks.* San Diego: IEEE.

Kanerva, P. (1988). *Sparse Distributed Memory.* Cambridge, MA: MIT Press.

Kolcz, A., & Allinson, N. (1994). Euclidian Input Mapping in a n-tuple approximation network. *Pages 285–289 of: Proceedings of the Sixth IEEE Digital Signal Processing Workshop.*

Luttrell, S. P. (1992). Gibbs distribution theory of adaptive n-tuple networks. *Pages 313–316 of:* Aleksander, I., & Taylor, J. (eds), *Artificial Neural Networks, 2.* Elsevier.

MacKay, D. (1992). Bayesian Interpolation. *Neural Computation*, **4**, 415–447.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (eds). (1994). *Machine Learning, Neural and Statistical Classification.* Prentice-Hall.

Rohwer, R. (1995). Two Bayesian treatments of the n-tuple recognition method. *Pages 171–176 of: Proc. IEE 4th International Conf. on Artificial Neural Networks (publication 409).*

Rohwer, R., & Cressy, D. (1989). Phoneme Classification by Boolean Networks. *Pages 557–560 of:* Tubach, J. P., & Mariani, J. J. (eds), *Proceedings of the European Conference on Speech Communication and Technology.* Paris: CEP, 26-28 Albany St., Edinburgh, Scotland.

Rohwer, R., & Lamb, A. (1993). An exploration of the effect of super large n-tuples on single layer RAMnets. *Pages 33 – 37 of:* Allinson, N. (ed), *Proceedings of the Weightless Neural Network Workshop '93, Computing with Logical Neurons.*

Sixsmith, M. J., Tattersall, G. D., & Rollett, J. M. (1990). Speech Recognition using n-tuple Techniques. *British Telecom Technology Journal*, **8**, 50–60.

Songcan, Y. Guoqing Ch., & Jun, L. (1992). Mulitilayer parallel distributed pattern recognition system model using sparse RAM nets. *IEE Proceedings-E*, **139**(2), 144–146.

Stonham, T.J. (1977). Improved Hamming–Distance Analysis for Digital Learning Networks. *Electronic Letters*, **13**(6), 155–156.

Sutton, R. S., & Whitehead, S. D. (1993). Online Learning with Random Representations. *In: Tenth International Conference on Machine Learning (ML 93).*

Tarling, Roland, & Rohwer, Richard. (1993). Efficient use of training data in the n-tuple recognition method. *Electronics Letters*, **29**(24), 2093–2094.

Tattersall, G. D., & Johnson, R. D. (1984). Speech recognisers based on N-tuple sampling. *Pages 405–413 of: Proc. Institute of Acoustics- Spring Conference.*

Tattersall, G. D., Foster, S., & Johnston, R. D. (1991). Single-layer lookup perceptrons. *IEE Proceedings-F*, **138**, 46–54.

Ullmann, J. R., & Kidd, P. A. (1969). Recognition Experiments with Typed Numerals from Envelopes in the Mail. *Pattern Recognition*, **1**, 273–289.

Ullmann, J.R. (1969). Experiments with the n-tuple Method of Pattern Recognition. *IEEE Transactions on Computers*, **18**(12), 1135–1137.

Figure 1 Tuple score vs. Hamming distance for a fixed test pattern $\boldsymbol{u}$ and training patterns $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_T\}$. The score of $\boldsymbol{u}$ was computed for the system trained on just one pattern $\boldsymbol{v}_i$. Distances between patterns of the same class are marked $*$, and $\circ$ is used for different classes. Functions (10) and (11) are plotted as 3 standard deviation error curves. The total number of patterns of the correct class (dotted line) and incorrect classes (solid line) are plotted in the margins as functions of Hamming distance (top) and tuple score (right). The means of these distributions are indicated by $*$ and $\circ$ marks. The "generalisation distance" L/n is indicated by a vertical dotted line.

Figure 2 **a)** actual network response to a test pattern $\boldsymbol{u}$; **b)** accumulated network response (as though patterns never overlapped on any tuples) and **c)** the number of new RAM cells addressed as a function of training patterns $\{\boldsymbol{v}_1, ..., \boldsymbol{v}_T\}$. Light lines are used to indicate the discriminator associated with the class that generated $\boldsymbol{u}$, dark lines denote the other class. The training data is sorted by the Hamming distance to the test pattern.

Figure 3 $1 + \frac{\log T}{-\log \epsilon}$ truncated to 0 for $T < \epsilon$ (dark lines) for $\epsilon = 0.25, 0.05, 0.005$ and 0.0001, and $\Theta_\theta(T)$ (light lines) for $\theta = 1, 2$ and 3, as functions of T. This shows that (25) can be a reasonable approximation at integral values of T in some circumstances, particularly for $\epsilon \to 0$ with $\theta = 1$.

Figure 4 Hamming distance between two CMAC/Gray-transformed integers vs. their arithmetic difference, for 3×10^4 randomly chosen pairs of integers. $K = 8$ and $a = 5$. The relationship is linear for Hamming distances up to K, and the transformed distance is bounded below by K for greater Hamming distances.

Figure 5 RAMnet's performance on Cut20 as a function of tuple size n for different values of a and K. The error bars of size $\pm$ 1 standard deviation are centred around the mean performance measured for 10 runs.

Figure 6 Results for N-tuple (●) and other algorithms.

Figure 7 Tuple score vs. Hamming Distance: **a)** Classification error (∗=952, ∘=970) due to high number of NN from the incorrect class; **b)** Correct classification (∗=803, ∘=564) resulting from the domination of NN patterns of the correct class; **c)** Classification error (∗= 805, ∘=884) caused by exceptionally large number of patterns from the incorrect class in the tail of the distribution (skewed priors); **d)** Correct classification (∗=996, ∘=994) thanks to the smaller tuple overlap on patterns from the correct class.

Figure 8 The number of hypercubes required to cover the space occupied by data. The datasets on which n-tuple classifier performed poorly are printed in bold face. A star denotes the existence of skewed priors.

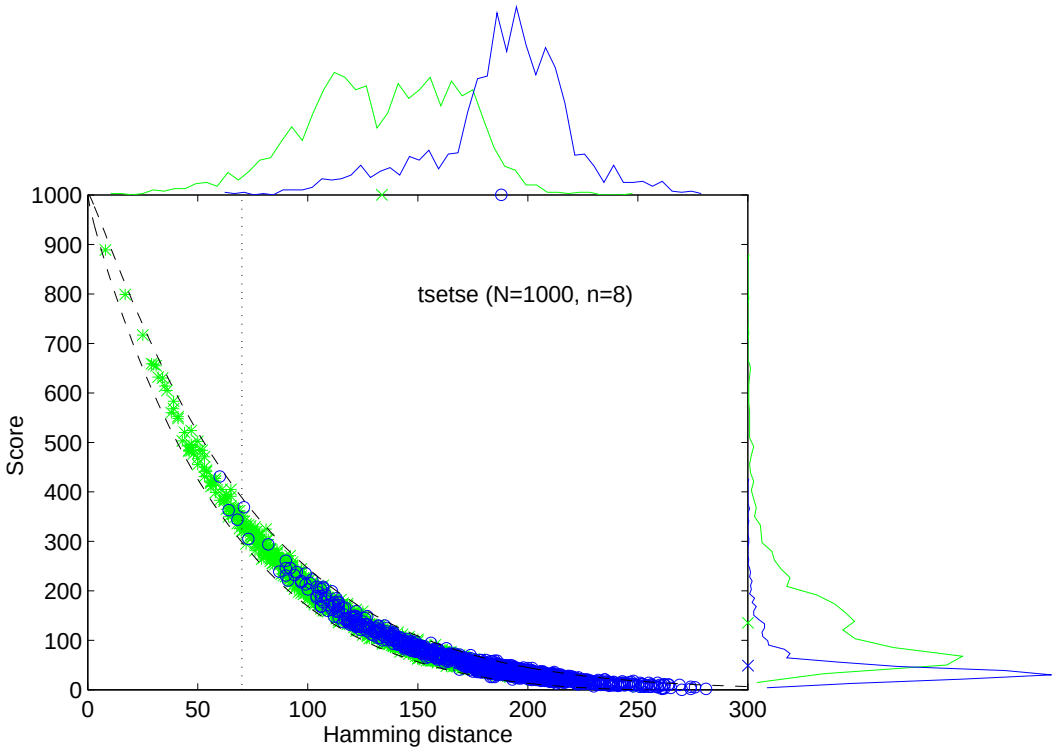
tsetse (N=1000, n=8)
Score
Hamming distance
0
100
200
300
400
500
600
700
800
900
1000
0
50
100
150
200
250
300

Figure 1:

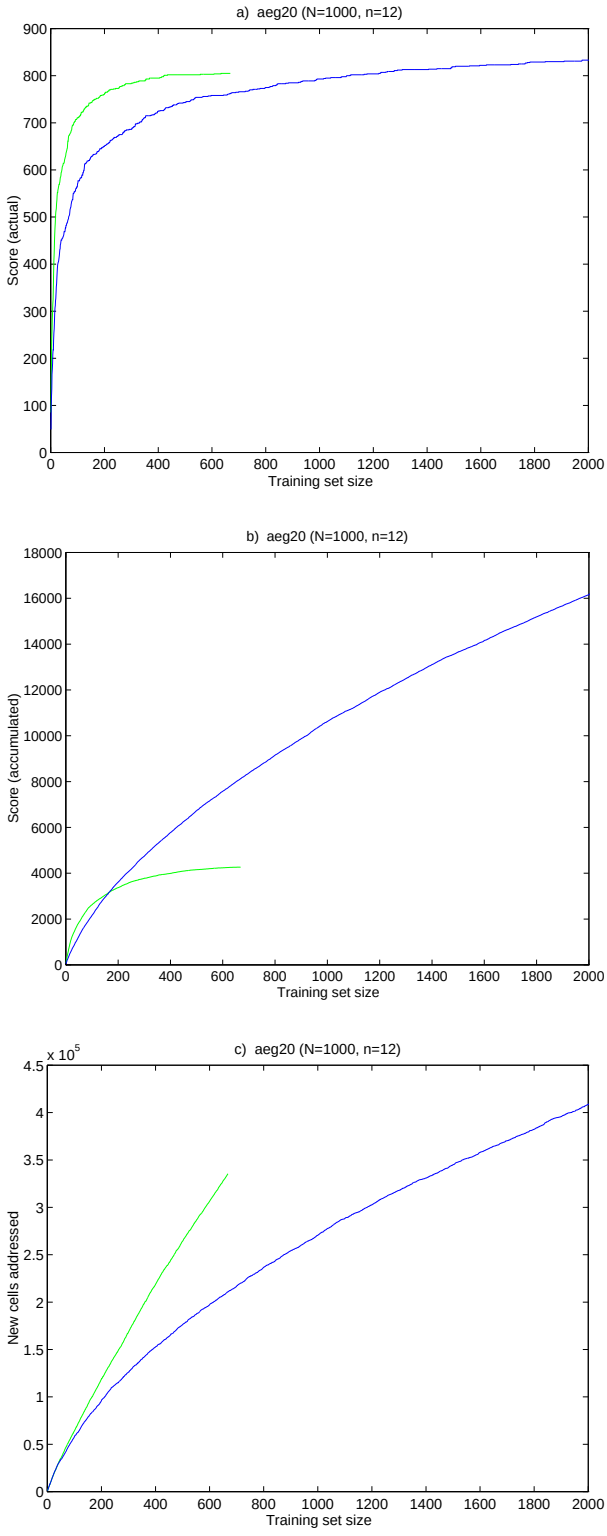
a) aeg20 (N=1000, n=12)
Score (actual)
Training set size
b) aeg20 (N=1000, n=12)
Score (accumulated)
Training set size
c) aeg20 (N=1000, n=12)
x 10^5
New cells addressed
Training set size

Figure 2:

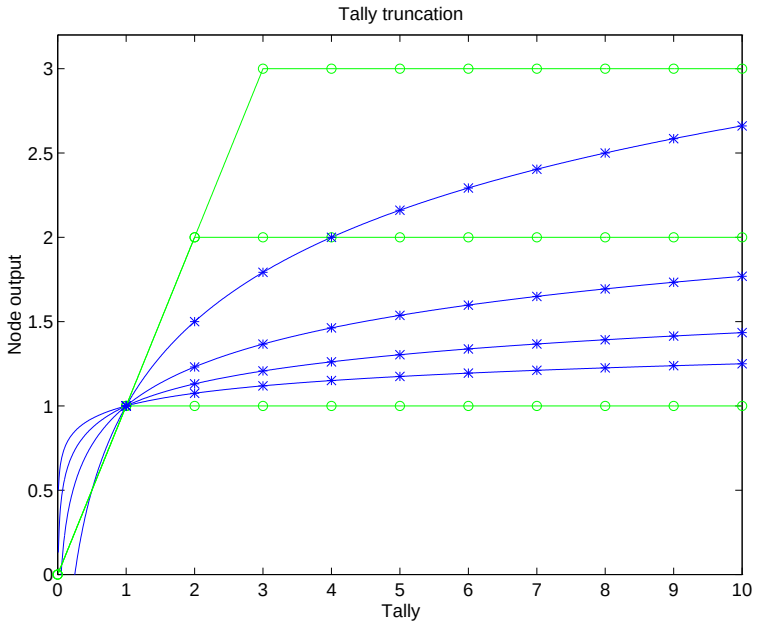

Figure 3:

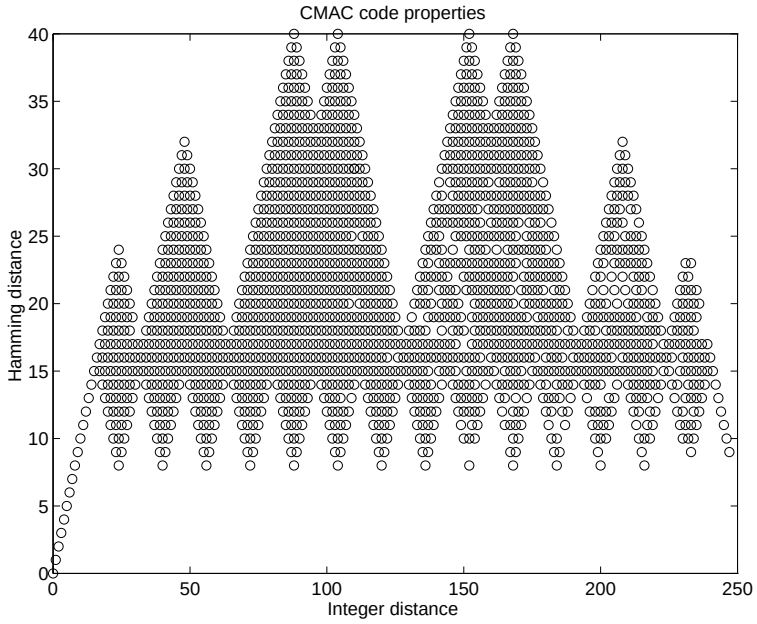

Figure 4:

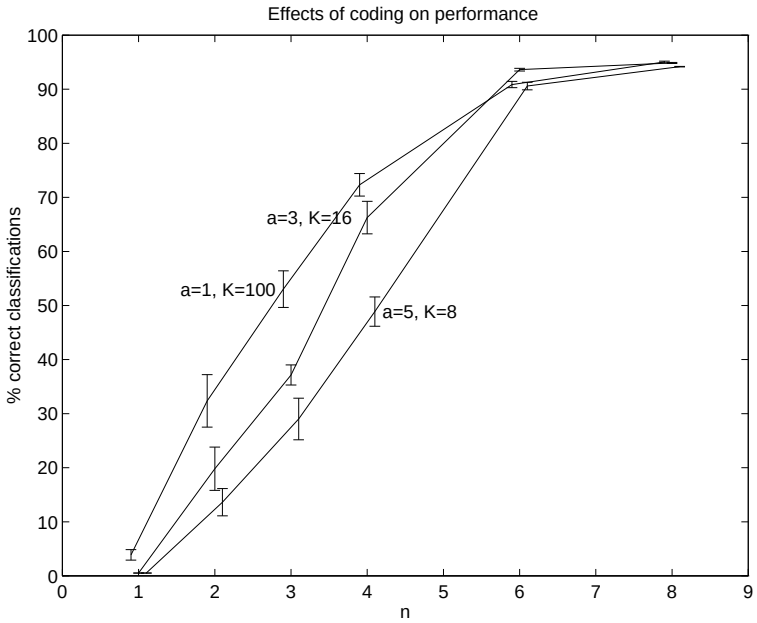
Effects of coding on performance
% correct classifications
n
a=1, K=100
a=3, K=16
a=5, K=8
0
10
20
30
40
50
60
70
80
90
100
0
1
2
3
4
5
6
7
8
9

Figure 5:

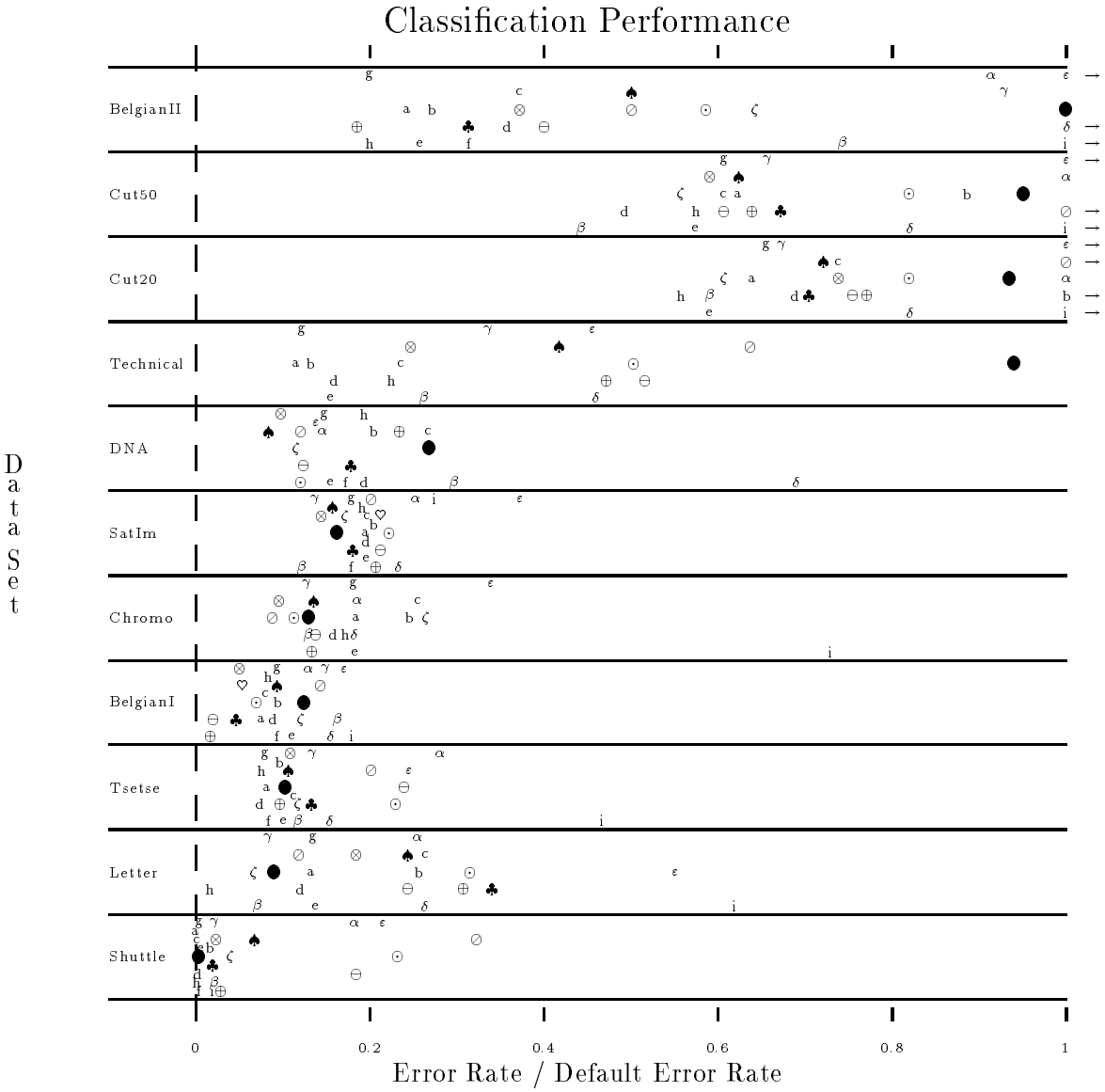
Classification Performance
BelgianII
Cut50
Cut20
Technical
DNA
SatIm
Chromo
BelgianI
Tsetse
Letter
Shuttle
Data Set
0
0.2
0.4
0.6
0.8
1
Error Rate / Default Error Rate

Figure 6:

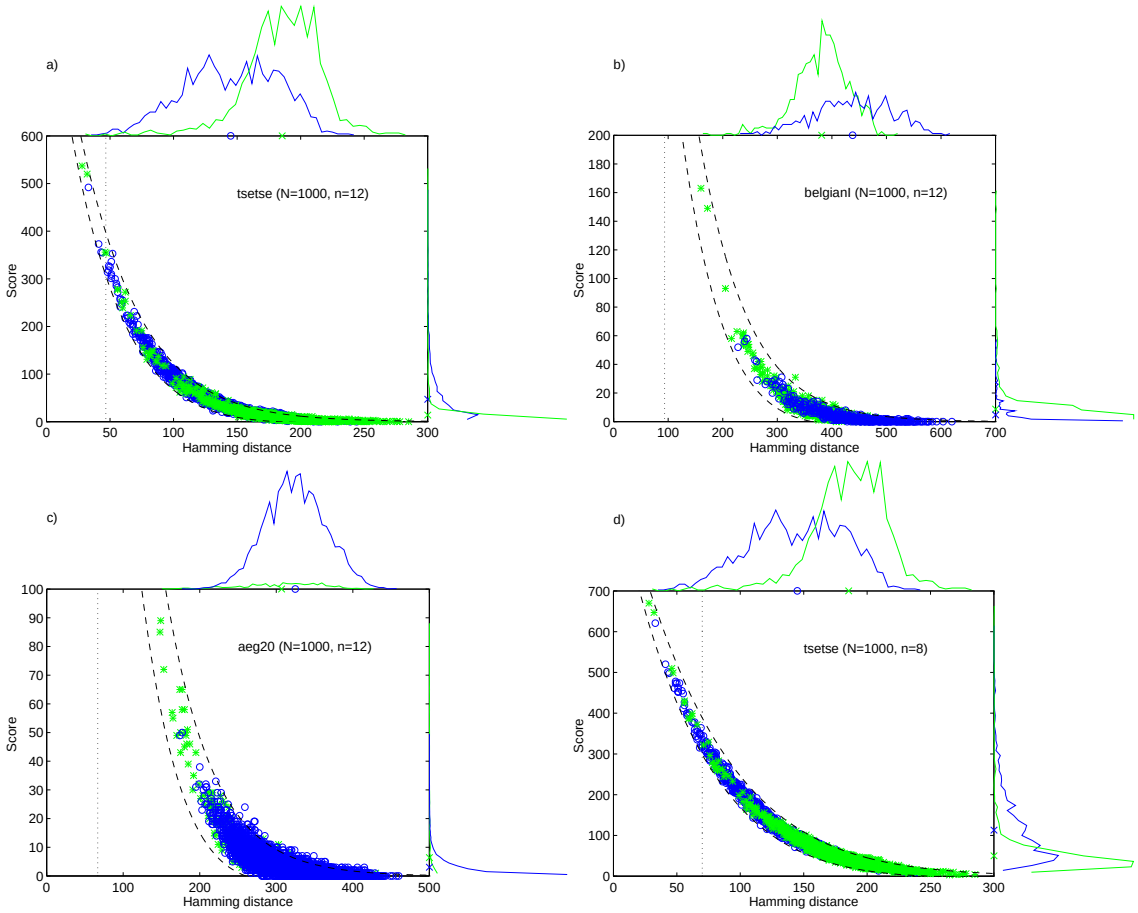

Figure 7:

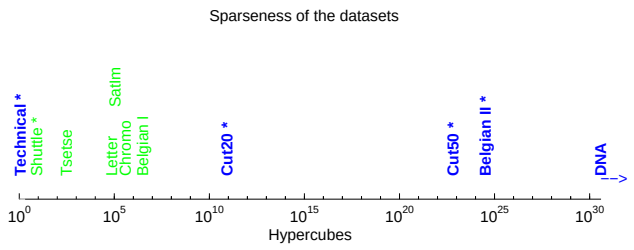

Figure 8:

RAMnets.

(●) n-tuple recogniser.

Discriminators.

(♣) Back Propagation in a 1-hidden-layer MLP.
(♠) Radial Basis Functions.
(♡) Cascade Correlation.
(⊕) SMART (Projection persuit).
(⊗) Dipol92 (based on pairwise linear discriminators).
(⊖) Logistic discriminant.
(⊘) Quadratic discriminant.
(⊙) Linear discriminant.

Methods related to density estimation.

(α) CASTLE (Probabilistic decision tree).
(β) k-NN (k nearest neighbours).
(γ) LVQ (Learning Vector Quantisation).
(δ) Kohonen topographic map.
(ε) Naive-Bayes (Estimate assuming independent attributes).
(ζ) ALLOC80 (Kernel function density estimator)

Decision trees.

(a) NewID (Decision Tree)
(b) AC^2 (Decision Tree)
(c) Cal5 (Decision Tree)
(d) CN2 (Decision Tree)
(e) C4.5 (Decision Tree)
(f) CART (Decision Tree)
(g) IndCART (CART variation)
(h) BayesTree (Decision Tree)
(i) ITrule (Decision Tree)

Table 1: Synopsis of Algorithms with symbols used in Figure 6.

Name	Classes	Largest Prior	Attributes	Training Patterns	Testing Patterns	Description
BelgianII	2	0.924	57 real	2000	1000	Classify measurements on simulated large scale power system as leading to stable or unstable behaviour.
Cut50	2	0.941	50 real	11220	7480	50 measurements from a candidate segmentation point in joined handwritten text. Classify as suitable cut point or not. Commercially confidential data.
Cut20	2	0.941	20 real	11220	7480	Best 20 attributes (by stepwise regression) from Cut50.
Technical	91	0.230	56	4500	2580	Commercially confidential. Appears to be generated by a decision tree. Most attribute values are 0.
DNA	3	0.525	180 Boolean	2000	1186	Sequences of 60 nucleotides (4-valued) classified into 3 categories.
SatIm	6	0.242	36 integer	4435	2000	3x3 pixel regions of Landsat images. Intensities in 4 spectral bands. Classified into 6 land uses at central pixel.
Chromo	24	0.044	16	20000	20000	Images of Chromosomes, reduced to 16 features.
BelgianI	2	0.5664	28 real	1250	1250	As Belgian II with a smaller simulation. Attributes thought to be least informative omitted from simulation.
Tsetse	2	0.508	14 real	3500	1499	Classify environmental attributes for presence of Tsetse flies.
Letter	26	0.045	16 16-valued	15000	5000	Images of typed capital letters, described by 16 real numbers discretised into 16 integers.
Shuttle	7	0.784	9 real	43500	14500	Classification problem concerning position of radiators on the Space Shuttle. Noise-free data.

Table 2: Descriptions of data sets used.