Two Bayesian treatments of the n-tuple recognition method

Richard Rohwer

NCRG/4323

Neural Computing Research Group
Technical Report No. NCRG/4323

May 1994

Neural Computing Research Group
Dept. of Computer Science and Applied Mathematics
Aston University
Aston Triangle
Birmingham B4 7ET
UK

Tel: +44 (0)21 359-3611
Fax: +44 (0)21 333-6215

Two Bayesian treatments of the n-tuple recognition method

Richard Rohwer
Dept. of Computer Science and Applied Mathematics
Aston University
Birmingham, UK B4 7ET

Abstract

Two probabilistic interpretations of the n-tuple recognition method are put forward in order to allow this technique to be analysed with the same Bayesian methods used in connection with other neural network models. Elementary demonstrations are then given of the use of maximum likelihood and maximum entropy methods for tuning the model parameters and assisting their interpretation.

1 Introduction

The n-tuple recognition method of Bledsoe and Browning [4, 2] is one of the oldest pattern recognition techniques which can be regarded as a neural network model. Practical experience has shown this classification method to be exceptionally fast and simple compared to more conventional methods, and usually similar in performance [7, 8]. However it does not enjoy firm theoretical underpinnings on a par with those provided to neural network methods which can be viewed from the standpoint of regression [6]. Here some basic formal machinery is laid out which makes this possible.

The n-tuple recognition method is defined in section 2, which is followed in section 3 by a terse review of the maximum likelihood method. These sections establish the notation for the rest of the paper. In the next two sections the n-tuple recognition method is wrapped in a probabilistic interpretation in two different ways, either of which can be used to arrive at a gradient descent method for training these systems with a maximum-likelihood objective. (These probabilistic interpretations are not to be confused with generalisations to stochastic n-tuple models [1]). A concluding section points out that this clears the way for the application of more accurate Bayesian methods such as Bayesian regularisation.

2 The n-tuple recognition method

The patterns to be classified are bit strings of a given length L. Several (let us say N) sets of n distinct[1] bit locations are selected randomly. These are the *n-tuples*. 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. 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.

Precisely, the class assigned to unclassified pattern $\boldsymbol{u}$ is

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

where $\mathcal{D}_c$ is the set of training patterns in class c, $\Theta(x) = 0$ for $x \leq 0$, $\Theta(x) = 1$ for $x > 0$, $\delta_{i,j}$ is the Kronecker delta[2] ($\delta_{i,j} = 1$ if $i = j$ and 0 otherwise.) and $\alpha_i(\boldsymbol{u})$ is the $i^{\underline{\text{th}}}$ feature of pattern $\boldsymbol{u}$:

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

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.

With C classes to distinguish, the system can be implemented as a network of NC nodes, each of which is a random access memory (RAM). The memory content $m_{ci\alpha}$ at address α of the $i^{\underline{\text{th}}}$ node allocated to class c is set to

$$m_{ci\alpha} = \Theta \left(\sum_{\boldsymbol{v} \in \mathcal{D}_c} \delta_{\alpha,\alpha_i(\boldsymbol{v})} \right). \tag{3}$$

Thus $m_{ci\alpha}$ is set if any pattern of $\mathcal{D}_c$ has feature α and unset otherwise. Recognition is accomplished by tallying the set bits in 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

$$\underset{c}{\operatorname{argmax}} \left(\sum_{i=1}^{N} m_{ci\alpha_i(\boldsymbol{u})} \right). \tag{4}$$

The method can be varied by changing Θ to a clipped ramp function with arbitrary threshold θ: $\Theta(x) = x$ for $x \leq \theta$, $\Theta(x) = \theta$ for $x > \theta$. For $\theta \to \infty$, the memory

[1] 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.

[2] The comma will be used optionally for extra clarity.

content $m_{ci\alpha}$ is simply the *tally*

$$T_{ci\alpha} = \sum_{\boldsymbol{v} \in \mathcal{D}_c} \delta_{\alpha, \alpha_i(\boldsymbol{v})} \tag{5}$$

Let us refer to all the $m_{ci\alpha}$ values collectively as $\boldsymbol{m}$. A corresponding notation using $\boldsymbol{T}$ will be used for tallies. Similarly, let $\boldsymbol{\eta}$ refer to all the *input mapping* values $\eta_i(j)$, with $\boldsymbol{\eta}_i$ referring to the n values defining the $i^{\underline{\text{th}}}$ n-tuple. An n-tuple recogniser is entirely specified by $\boldsymbol{\eta}$ and $\boldsymbol{m}$. Here $\boldsymbol{\eta}$ will be considered given, so that $\boldsymbol{m}$ will suffice to specify a single recogniser.

3 Maximum likelihood

This section establishes some notation in the context of a brief review of the maximum likelihood method, and related methods of selecting model parameters. It is assumed that there is an unknown probability density[3] $P(c, \boldsymbol{u})$ for being presented with a pattern $\boldsymbol{u}$ from class c. (The classes are not necessarily taken to be mutually exclusive; the same pattern can have non-zero probability in more than one class.) The marginal and conditional distributions related to $P(c, \boldsymbol{u})$ will be denoted in the standard way. There is a parameterised class of models intended to approximate $P\left(c \,\middle|\, \boldsymbol{u}\right)$, the model with parameters $\boldsymbol{m}$ being denoted $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right)$. Typically, a model $P\left(\boldsymbol{u} \,\middle|\, c; \boldsymbol{m}\right)$ of $P\left(\boldsymbol{u} \,\middle|\, c\right)$ is more readily available, in which case $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right)$ is obtained from Bayes' rule

$$P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right) = \frac{P\left(\boldsymbol{u} \,\middle|\, c; \boldsymbol{m}\right) P(c)}{\sum_{c'} P\left(\boldsymbol{u} \,\middle|\, c'; \boldsymbol{m}\right) P(c')} \tag{6}$$

and an estimate of $P(c)$. (The sum over all possible patterns would be replaced by an integral in applications involving real-valued patterns.) In any case, the general idea is to guess which model $\boldsymbol{m}^*$ will approximate $P(c, \boldsymbol{u})$ best, on the basis of information provided by a training sample $\mathcal{D}$ of patterns generated by $P(c, \boldsymbol{u})$, and to then use $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}^*\right)$ to classify unknown patterns. (This approximates a more correct approach involving a sum over an estimated distribution of model probabilities.)

The optimal model $\boldsymbol{m}^*$ can be obtained by maximising over a probability distribution over models, which in turn is obtained from the training data and a prior distribution $P^0(\boldsymbol{m})$ expressing a guess of how likely each model should be thought

[3] Throughout this paper we shall engage in the technically dubious but conventional practice of using the names of the variables in a probability density to also designate which density is meant. This implies that variables cannot be renamed arbitrarily. We shall avoid ambiguity through the convention of generating new variable names only by appending primes ($'$).

to be if $\mathcal{D}$ were unknown. Bayes' rule again provides the required expression

$$P\left(\boldsymbol{m} \,\middle|\, \mathcal{D}\right) = \frac{P\left(\mathcal{D} \,\middle|\, \boldsymbol{m}\right) P^0(\boldsymbol{m})}{\sum_{m'} P\left(\mathcal{D} \,\middle|\, \boldsymbol{m}'\right) P^0(\boldsymbol{m}')} \tag{7}$$

in which an assumption of independent data samples provides:

$$P\left(\mathcal{D} \,\middle|\, \boldsymbol{m}\right) = \prod_c \prod_{\boldsymbol{u} \in \mathcal{D}_c} P\left(c, \boldsymbol{u} \,\middle|\, \boldsymbol{m}\right). \tag{8}$$

Here the training data $\mathcal{D}$ is regarded as split up into a set of subsets $\mathcal{D}_c$ of patterns from each class.

In (8), $P\left(c, \boldsymbol{u} \,\middle|\, \boldsymbol{m}\right)$ can be expressed either as $P\left(\boldsymbol{u} \,\middle|\, c; \boldsymbol{m}\right) P(c)$ or $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right) P(\boldsymbol{u})$. Typically both factors in the first form are easier to estimate than the corresponding factors of the second form, but it will be seen shortly that the n-tuple recogniser can lend itself to the atypical treatment.

Under a uniform prior $P^0(\boldsymbol{m})$, $P\left(\boldsymbol{m} \,\middle|\, \mathcal{D}\right)$ is proportional to the *likelihood* $P\left(\mathcal{D} \,\middle|\, \boldsymbol{m}\right)$, in which case $\boldsymbol{m}^*$ is the maximum-likelihood model.

4 First probabilistic interpretation

In order to commence with a maximum-likelihood treatment, a probabilistic interpretation of n-tuple recognisers must be provided; *ie.*, the form of either $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right)$ or $P\left(\boldsymbol{u} \,\middle|\, c; \boldsymbol{m}\right)$ must be specified for use in (8) and then (7). Any interpretation will do, provided it is used consistently at each step of the procedure and provided there exist parameters yielding an accurate approximation to the true distribution, $P\left(c \,\middle|\, \boldsymbol{u}\right)$ or $P\left(\boldsymbol{u} \,\middle|\, c\right)$.

One natural interpretation is suggested by the usage (4) of $\boldsymbol{m}$ in classifying unknown patterns:

$$P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right) = \sum_i m_{ci\alpha_i(\boldsymbol{u})} \Big/ \sum_{c'i} m_{c'i\alpha_i(\boldsymbol{u})} \tag{9}$$

A prior $P^0(\boldsymbol{m})$ which favours setting all $m_{ci\alpha}$ equal is sensible with a random input mapping $\boldsymbol{\eta}$, because there is nothing *a priori* that can be said about one memory location that cannot be said about another. But it is simpler and not wildly unreasonable to use a prior which is uniform over all arrangements of memory contents $\boldsymbol{m}$, because then (7) submits to a maximum likelihood treatment. Plugging (9) into (8) using $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right) P(\boldsymbol{u})$ for $P\left(c, \boldsymbol{u} \,\middle|\, \boldsymbol{m}\right)$ gives

$$P\left(\mathcal{D} \,\middle|\, \boldsymbol{m}\right) = \prod_c \prod_{\boldsymbol{u} \in \mathcal{D}_c} \left(\sum_i m_{ci\alpha_i(\boldsymbol{u})} \Big/ \sum_{c'i} m_{c'i\alpha_i(\boldsymbol{u})} \right) P(\boldsymbol{u}) \tag{10}$$

for the likelihood. Because the logarithm is monotonicly increasing, the most likely model is the maximum with respect to $\boldsymbol{m}$ of the log likelihood

$$I\left(\mathcal{D}\,\middle|\,\boldsymbol{m}\right)=\sum_{c}\sum_{\boldsymbol{u}\in\mathcal{D}_c}\left(\ln\left(\sum_i m_{ci\alpha_i(\boldsymbol{u})}\right)-\ln\left(\sum_{c'i} m_{c'i\alpha_i(\boldsymbol{u})}\right)+\ln\left(P\left(\boldsymbol{u}\right)\right)\right). \tag{11}$$

The final term $\ln\left(P\left(\boldsymbol{u}\right)\right)$ can be omitted because it does not depend on $\boldsymbol{m}$.

The extrema of the log likelihood $I\left(\mathcal{D}\,\middle|\,\boldsymbol{m}\right)$ satisfy $dI\left(\mathcal{D}\,\middle|\,\boldsymbol{m}\right)/dm_{c''i''\alpha''}=0$, or

$$\sum_{\boldsymbol{u}\in\mathcal{D}_{c''}}\frac{\delta_{\alpha'',\alpha_{i''}(\boldsymbol{u})}}{\sum_i m_{c''i\alpha_i(\boldsymbol{u})}}-\sum_c\sum_{\boldsymbol{u}\in\mathcal{D}_c}\frac{\delta_{\alpha'',\alpha_{i''}(\boldsymbol{u})}}{\sum_{c'i} m_{c'i\alpha_i(\boldsymbol{u})}}=0. \tag{12}$$

The denominator of the first term is the total number of memory locations for class c addressed by pattern $\boldsymbol{u}$. Abbreviating this *class c weight in memory* of pattern $\boldsymbol{u}$ as

$$M_c(\boldsymbol{u})=\sum_i m_{ci\alpha_i(\boldsymbol{u})}, \tag{13}$$

and similarly abbreviating the *total weight in memory* of pattern $\boldsymbol{u}$ as

$$M(\boldsymbol{u})=\sum_{ci} m_{ci\alpha_i(\boldsymbol{u})}, \tag{14}$$

(12) can be written

$$\sum_{\boldsymbol{u}\in\mathcal{D}_c}\frac{\delta_{\alpha\alpha_i(\boldsymbol{u})}}{M_c(\boldsymbol{u})}=\sum_{\boldsymbol{u}\in\mathcal{D}}\frac{\delta_{\alpha\alpha_i(\boldsymbol{u})}}{M(\boldsymbol{u})}. \tag{15}$$

This says that for each memory location, the sum of the inverse class weights of all patterns which address that location must be the same for every class, and that this common value is the corresponding sum of inverse total weights.

Although (15) is a system of $N2^n$ linear equations in the CD variables $1/M_c(\boldsymbol{u})$ (where D is the total number of training patterns), the variables are related by (13), so a method of solution is not obvious. There could be as few as N non-trivial equations in (15) if for every i, $\alpha_i(\boldsymbol{u})$ were the same for any pattern $\boldsymbol{u}$ in the training data. Because (10) and therefore (11) is bounded above, there must be a solution, and furthermore there must be at least a 1-parameter family of solutions because neither (10) nor (15) is affected by multiplication of all the $m_{ci\alpha}$ by a constant scale factor.

In particular there is no reason to suppose that the conventional prescription (3), which gives

$$M_c(\boldsymbol{u})=\sum_i\Theta\left(\sum_{\boldsymbol{v}\in\mathcal{D}_c}\delta_{\alpha_i(\boldsymbol{u}),\alpha_i(\boldsymbol{v})}\right) \tag{16}$$

provides a solution. If real values were admissible for memory contents, then it would be possible to improve on (3) by gradient ascent of (11) from such an initial guess. In its simplest form, the algorithm would repeatedly make the replacement

$$m_{ci\alpha}\leftarrow m_{ci\alpha}+\epsilon\left(\sum_{\boldsymbol{u}\in\mathcal{D}_c}\frac{\delta_{\alpha\alpha_i(\boldsymbol{u})}}{M_c(\boldsymbol{u})}-\sum_{\boldsymbol{u}\in\mathcal{D}}\frac{\delta_{\alpha\alpha_i(\boldsymbol{u})}}{M(\boldsymbol{u})}\right) \tag{17}$$

where ϵ is a small real number, until a solution was found. If the memory were restricted to integer or binary values, then a global stochastic maximisation algorithm such as simulated annealing or a genetic algorithm could be applied to find the maxima of (11).

5 Second probabilistic interpretation

Another probabilistic interpretation of $\boldsymbol{m}$ can be motivated by the training rule (3). By recording which n-tuples occur in the training data, $\boldsymbol{m}$ provides some information about the marginal distributions

$$P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c\right) = \sum_{u_j \notin \boldsymbol{\eta}_i} P\left(\boldsymbol{u} \,\middle|\, c\right) \tag{18}$$

The indexing on the sum is meant to indicate a sum over all components of $\boldsymbol{u}$ except those in the range of $\boldsymbol{\eta}_i$, and $\boldsymbol{u}_{\boldsymbol{\eta}_i}$ indicates the subpattern $\{u_j | j \in \boldsymbol{\eta}_i\}$. Further to footnote 3, the i in $P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c\right)$ identifies which marginal distribution is meant. More precisely,

$$P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c\right) = \sum_{\boldsymbol{u}'} \prod_{j=0}^{n-1} \delta_{u_{\eta_i(j)} u'_{\eta_i(j)}} P\left(\boldsymbol{u}' \,\middle|\, c\right) \tag{19}$$

with the $\boldsymbol{u}'$ sum ranging over all possible patterns, which is to say

$$P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c\right) = \sum_{\boldsymbol{u}'} \delta_{\alpha_i(\boldsymbol{u})\alpha_i(\boldsymbol{u}')} P\left(\boldsymbol{u}' \,\middle|\, c\right). \tag{20}$$

Note that although the final expression is written in terms of all the components of $\boldsymbol{u}$, it depends only on those in $\boldsymbol{\eta}_i$.

With infinite threshold, training prescription (3) gives $\boldsymbol{m} = \boldsymbol{T}$, in which case $\boldsymbol{m}$ can be interpreted as estimates of the marginal probabilities

$$P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c; \boldsymbol{m}\right) = m_{ci\alpha_i(\boldsymbol{u})} / \sum_{\alpha} m_{ci\alpha}. \tag{21}$$

The maximum entropy distribution under these constraints might reasonably serve to define $P\left(\boldsymbol{u} \,\middle|\, c; \boldsymbol{m}\right)$ for use in (8). If the tallies are thresholded to give $\boldsymbol{m}$, the constraint becomes more complicated; eg., for $\theta = 1$, (21) could be replaced by

$$\begin{aligned} P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c; \boldsymbol{m}\right) &< 1/D_c \qquad && m_{ci\alpha_i(\boldsymbol{u})} = 0 \\ P\left(\boldsymbol{u}_{\boldsymbol{\eta}_i} \,\middle|\, c; \boldsymbol{m}\right) &> 1/D_c \qquad && m_{ci\alpha_i(\boldsymbol{u})} = 1 \end{aligned} \tag{22}$$

where D_c is the number of training samples for class c.

Due to sampling fluctuations, (21) or (22) will not be precisely correct statements about the true $P\left(\boldsymbol{u} \,\middle|\, c\right)$. Some improvement might be obtained by carrying through the maximum likelihood prescription to determine $\boldsymbol{m}$, rather than following (3).

Whether or not the maximum entropy distribution is used for this purpose, it is needed in (6) to obtain the distribution $P\left(c \,\middle|\, \boldsymbol{u}; \boldsymbol{m}\right)$ from which classification decisions are made. These decisions will not necessarily turn out to be equivalent to (4). Here the programme will be examined to the extent of obtaining expressions for the maximum entropy distribution, and discussing a simple special case. Under simplifying assumptions to be examined shortly, (3) results anyway.

Let us consider the maximum entropy problem under the simpler equality constraints (21). With

$$h_{ci\alpha} = m_{ci\alpha} / \sum_{\alpha'} m_{ci\alpha'}, \tag{23}$$

the constraints (21) can be written

$$\sum_{\boldsymbol{u}} \delta_{\alpha\alpha_i(\boldsymbol{u})} P\left(\boldsymbol{u} \,\middle|\, c\right) = h_{ci\alpha} \tag{24}$$

because with an arbitrary pattern $\boldsymbol{u}$, $\alpha_i(\boldsymbol{u})$ simply refers to an arbitrary address. There is also a normalisation constraint, $\sum_{\boldsymbol{u}} P\left(\boldsymbol{u} \,\middle|\, c\right) = 1$.

Introducing Lagrange multipliers $\beta_{ci\alpha}$ and γ_c for these constraints, a standard maximum entropy treatment leads to

$$P\left(\boldsymbol{u} \,\middle|\, c; \boldsymbol{m}\right) = \frac{1}{Z_c} e^{-\sum_i \beta_{ci\alpha_i(\boldsymbol{u})}} \tag{25}$$

with the multiplier γ_c absorbed into the normalisation factor $Z_c = \sum_{\boldsymbol{u}} e^{-\sum_i \beta_{ci\alpha_i(\boldsymbol{u})}}$ and the β's related to the h's by

$$\sum_{\boldsymbol{u}} \delta_{\alpha\alpha_i(\boldsymbol{u})} \frac{1}{Z_c} e^{-\sum_k \beta_{ck\alpha_k(\boldsymbol{u})}} = h_{ci\alpha}. \tag{26}$$

This can be simplified further if the n-tuples never overlap, ie., for all i and j, $\boldsymbol{\eta}_i \bigcap \boldsymbol{\eta}_j = \emptyset$. Then (26) becomes a product of sums which reduces after a little algebra to

$$e^{-\beta_{ci\alpha}} = m_{ci\alpha}. \tag{27}$$

This result is unsurprising. If the n-tuples do not overlap, then in the absence of further information about the distribution of patterns, nothing requires the marginal distributions for different n-tuples to be correlated, so the maximum entropy distribution is simply the product of the marginals. Result (27) followed from the fact that (25) is readily broken up into such a product. This has been noted previously by Luttrell [5], and the independence assumption has led to good results in some applications [3]. However, there would normally be prior information which puts this assumption in doubt. The similarities between the patterns of a class are bound to involve correlations between their constituent bits.

An alternative approach, more complicated but better motivated, is to add further constraints to the problem to express more than (21) of what is known either *a* *priori* or from the training data. Introducing Lagrange multipliers for these constraints leads to generalisations of (25) in the standard manner. It is possible to use the assumption of non-overlapping n-tuples to disentangle the sums, through the device of replacing each pattern bit with a set of identically set bits, each of which participates in at most one n-tuple. Of course, constraints should then be introduced which express the resulting correlations.

It is a straightforward matter to obtain equations for the most likely $\boldsymbol{m}$ under the maximum entropy distribution. In general there is no obvious method of solution, but under the independence assumption with no additional constraints, simplifications similar to those leading to (27) result in a solution agreeing with (3).

6 Conclusions

Two probabilistic interpretations of the n-tuple recognition method have been presented and discussed. The first is technically simpler, in that class probabilities are given directly by the model, and the second is more conventional, in that the model produces pattern probabilities. Either way, the machinery of Bayesian methods can be applied. Here this has been carried out only in the simplest, maximum likelihood approximation. There appears to be no particular barrier to more complicated and realistic treatments, incorporating prior knowledge through regularisation methods, or via constraints in a maximum entropy problem. This should lead to improvements upon and better understanding of this old, unusual, and surprisingly effective neural network model.

References

[1] I. Aleksander. The logic of connectionist systems. In I. Aleksander, editor, *Neural computing architectures*, pages 133–155. MIT Press, 1989.

[2] I. Aleksander, W. V. Thomas, and P. A. Bowden. Wisard: A radical step forward in image recognition. *Sensor Review*, 4:120–124, 1984.

[3] Ardeshir Badr. N-tuple classifier for ecg signals. In N. Allinson, editor, *Proceedings of the Weightless Neural Network Workshop '93, Computing with Logical Neurons*, pages 29 – 32, University of York, 1993.

[4] W. W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Proceedings of the Eastern Joint Computer Conference*, pages 232–255, Boston, 1959.

[5] S. P. Luttrell. Gibbs distribution theory of adaptive n-tuple networks. In I. Aleksander and J. Taylor, editors, *Artificial Neural Networks, 2*, pages 313–316. Elsevier, 1992.

[6] D. MacKay. Bayesian interpolation. *Neural Computation*, 4:415–447, 1992.

[7] R. Rohwer and D. Cressy. Phoneme classification by boolean networks. In J. P. Tubach and J. J. Mariani, editors, *Proceedings of the European Conference on Speech Communication and Technology*, pages 557–560, Paris, 1989. CEP, 26-28 Albany St., Edinburgh, Scotland.

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