Theoretical Analysis and Improved Decision Criteria for the n-Tuple Classifier

Thomas Martini Jørgensen and Christian Linneberg

Abstract—The anticipated behavior of the n-tuple classification system is that it gives the highest output score for the class to which the input example actually belongs. By performing a theoretical analysis of how the output scores are related to the underlying probability distributions of the data, this paper shows that this in general is not to be expected. The theoretical results are able to explain the behavior that is observed in experimental studies. The theoretical analysis also give valuable insight into how the n-tuple classifier can be improved to deal with skewed training priors, which until now have been a hard problem for the architecture to tackle. It is shown that by relating an output score to the probability that a given class generates the data makes it possible to design the n-tuple net to operate as a close approximation to the Bayes estimator. It is specifically illustrated that this approximation can be obtained by modifying the decision criteria. In real cases, the underlying example distributions are unknown and accordingly the optimum way to treat the output scores cannot be calculated theoretically. However, it is shown that the feasibility of performing leave-one-out cross-validation tests in n-tuple networks makes it possible to obtain proper processing of the scores in such cases.

Index Terms—n-tuple classifier, maximum likelihood, Bayes, cross-validation, RAM-net.

1 Introduction

The so-called n-tuple classifier was invented as a pattern recognition device in 1959 by Bledsoe and Browning [6]. In the learning phase, the n-tuple classifier stores class-specific information about the training set in a number of look-up tables. The entries in each look-up table are addressed by sampling n specific data locations (forming an n-tuple) of the input presented to the classifier. When performing a classification, all addressed entries in the look-up tables are summed class-wise and a winner-takes-all (WTA) decision is made to classify the input vector. As the standard n-tuple net is based on one-shot learning it is very fast to train compared with other classification methods. Furthermore, the architecture is highly feasible for hardware implementations [3], [4], [12].

Early on, different strategies for optimising the choices of data points to be sampled by the n-tuples were studied [7] and many modifications and improvements of the architecture have been described in the literature, where the architectures are often denoted weightless neural networks or RAM-based neural networks [5]. Valuable insight into the dependency between the output scores of the n-tuple network and the Hamming distance between training and test examples has also been provided at an early phase. Thus in 1970, Aleksander presented an approximate formula describing the relationship between Hamming distances and output scores [1]. Stonham later modified this formula [17] to incorporate the overlap between training examples, which plays a crucial role in understanding the behavior of the n-tuple net. However, a more thorough analysis of the n-tuple architecture based on these derivations has been lacking due to the difficulty in dealing with the combinatorial explosion found in the mathematical approaches used in analysing the architecture [17], [5].

There has also been an effort in putting the mechanism of the so-called frequentist n-tuple classifier into a Bayesian framework [14]. The approach is based on the fact that naïve estimates of conditional feature estimates arise naturally within the n-tuples. However, sub-patterns that never appear in the training set lead to zero estimates although these patterns might occur for examples outside the training set. By replacing these zero estimates with a small ad hoc constant, crude maximum likelihood estimators are obtained. In the limit with a very small value of the constant, this corresponds to the winner-takes-all classification scheme. Although the work actually places the n-tuple classifier in a Bayesian framework by interpreting the contents of the n-tuples as probability densities, the interpretation is only a valid approach if the contents of the n-tuples closely reflect the true distributions of the data. In order to understand the fundamental behavior of the n-tuple classifier, it is needed to find the relations between the output scores and the true distributions.

A recent comprehensive experimental benchmark study of the n-tuple classifier performed by Rohwer and Morciniec documents very well the advantages as well as problems with the standard n-tuple classifier [16]. In spite of the simplicity of the architecture and the training procedure the benchmark results show that the n-tuple classifier in 6 out of 11 data sets were competitive with the 23 other classification methods used in the benchmark study. On the remaining five data sets, the n-tuple classifier performed poorly compared to most of the other methods. One characteristic of these five data sets is that the number of training examples varies a lot from one class to the other; this

- *T.M. Jørgensen is with Risø National Laboratory, P.O. Box 49, DK-4000 Roskilde, Denmark. E-mail: thomas.martini@risoe.dk.*
- *C. Linneberg is with Intellix A/S, H.S. Ørstedsvej 4, DK-1879 Copenhagen, Denmark. E-mail: cli@intellix.com.*

Manuscript received 28 May 1998. Recommended for acceptance by A. Webb. For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107817.

feature is often denoted skewed priors. An essential weakness of the standard n-tuple architecture is simply its general lack of ability to deal satisfactorily with such priors.

In the present paper we present a theoretical analysis that relates the expected output scores of the n-tuple net to the stochastic parameters of the example distributions, the number of available training examples, and the number of address lines n used for each n-tuple. From the obtained expressions, we are able to study the behavior of the architecture in different scenarios. Furthermore, it becomes possible to deduce how the n-tuple classifier should be modified to operate as a close approximation to the maximum a posteriori or maximum likelihood estimator. The resulting modified decision criteria can for instance deal with the problems caused by skewed training priors, and accordingly they provide an essential improvement of the architecture. Finally, we illustrate how the feasibility of performing a leave-one-out cross-validation test for the n-tuple architecture plays an essential role in obtaining the new decision borders on real data.

The suggested changes in decision criteria are not only applicable to the original n-tuple architecture based on random memorization. It also applies to extended n-tuple schemes, some of which use a more optimal selection of the address lines (see e.g., the architectures described in [8] and [9]).

2 The n-Tuple Classifier

The n-tuple classifier is a simple memory based classifier [2], [6]. One of the major benefits of a memory-based system is its very fast computation time, both during the learning phase and during classification. For n-tuple networks, which are also known as "RAM networks" or "weightless neural networks" [5], learning may be accomplished by recording features of patterns in a random-access memory (RAM), which requires just one presentation of the training set to the system.

The n-tuple classifier can be considered as a number of look-up tables (LUTs). The number of rows in each LUT corresponds to the number of possible classes. Which columns to address in the tables are determined from the values at the input connections of the LUTs. The input values to a LUT are given as a subset of the data values defining a given example. The data values are normally represented by a bit string. With the standard n-tuple classifier, the subsets to be used are selected at random. The values sampled by a given LUT constitute a specific feature of the presented example.

Before training, all entries of all LUTs are set to zero. During training, all examples are presented to the classifier. The content of a given cell in a LUT is set to one if it is addressed by a training example. The class of the training example determines the row of the addressed cell and as described above the sampled data values determine the column.

When performing a classification all values of the addressed columns are read out. A value of one corresponds to a vote on the class specified by the row value of the addressed cell. The votes from all LUTs are summed to obtain the output score for each class. Subsequently a winner-takes-all decision is made; i.e., a pattern is classified as belonging to the class where most of the sampled features in the pattern are shared with those found in the training examples from that class.

More rigorously, consider a training set $\boldsymbol{X}$ and training examples $\overline{x}_i \in \boldsymbol{X}$. Then the class, $\mathrm{C}(\overline{y})$, assigned to an example $\overline{y}$ presented to the n-tuple classifier can be calculated as

$$\mathrm{C}(\overline{y}) = \underset{c}{\operatorname{argmax}}\left(\mathrm{S}_c\left(\boldsymbol{X}_c, \overline{y}\right)\right), \tag{1}$$

where $\mathrm{S}_c(\cdot)$ denotes the relative output score for class c and $\boldsymbol{X}_c$ is the set of training examples labelled class c. The relative output score denotes the number of votes obtained for the class in question relative to the number of LUTs. For the standard n-tuple classifier with binary input values, the address function $a_i(\overline{y})$ indexes the column in the ith LUT being addressed by example $\overline{y}$. It is normally calculated as

$$\mathrm{a}_\mathrm{i}(\overline{y}) = \sum_{j=1}^{n} y_{\psi_{ij}} 2^{j-1}. \tag{2}$$

Here ψ_{ij} is the jth address line of the ith LUT forming an index into the example $\overline{y}$, and n denotes the number of address lines per LUT. Let $v_{\mathrm{a_i}(\overline{y}),c}$ represent the number of training examples accessing the LUT cell addressed in the ith LUT by the example $\overline{y}$ and the class c:

$$v_{\mathrm{a_i}(\overline{y}),c} = \sum_{\overline{x} \in \boldsymbol{X}_c} \delta_{\mathrm{a_i}(\overline{x}),\mathrm{a_i}(\overline{y})}. \tag{3}$$

Here $\delta_{i,j}$ denotes Kroneckers delta ($\delta_{i,j} = 1$ if $i = j$ and $\delta_{i,j} = 0$ if $i \neq j$). With these definitions $\mathrm{S_c}(\cdot)$ can be written as:

$$\mathrm{S_c}\left(\boldsymbol{X}_c, \overline{y}\right) = \frac{1}{N_\Omega} \sum_{i \in \Omega} \Theta_k\left(v_{\mathrm{a_i}(\overline{y}),c}\right), \tag{4}$$

where Ω describes the set of N_Ω LUTs making up the whole network. $\Theta_k(z)$ is the step function with an offset of k:

$$\Theta_k(z) = \begin{cases} 1 & \text{if } z \geq k \\ 0 & \text{if } z < k \end{cases}. \tag{5}$$

Using $k = 1$ in (4) gives the "vanilla" n-tuple classifier, which we will address in this paper.

The generalization capability of the network is directly related to the number of address lines (this number n does not need to be the same for all LUTs, although this often has been the case in many implementations). If a LUT samples all input bits then it will act as a pure memory device with no generalization capabilities. As the number of input bits is reduced, the generalization is increased at expense of an increasing number of ambiguous decisions. Furthermore, the classification and generalization performances of a LUT are highly dependent on the actual subset of input bits probed.

Estimating the generalization capabilities of a classifier can be done by performing a leave-one-out cross-validation where in turn a single example is left out of the training set; i.e., the classifier is trained on the reminder of the training set and tested on the one example excluded from the training set. As pointed out by Liisberg and described in [10] it

is easy to incorporate a leave-one-out cross-validation test for the n-tuple classifier. The leave-one-out cross-validation classification of a training example $\overline{x}$ can be calculated as

$$\Lambda(\overline{x}) = \underset{c}{\operatorname{argmax}}\left(\sum_{i\in\Omega}\Theta_{1+\delta_{C_T(\overline{x}),c}}\left(v_{a_i(\overline{x}),c}\right)\right), \tag{6}$$

where $C_T(\overline{x})$ denotes the true class of example $\overline{x}$. With $N_{\boldsymbol{X}}$ denoting the number of training examples the total cross-validation error rate is given by

$$1-\frac{1}{N_{\boldsymbol{X}}}\sum_{\overline{x}_i\in\boldsymbol{X}}\delta_{\Lambda(\overline{x}_i),C_T(\overline{x}_i)}. \tag{7}$$

Comparing (6) with (1) and (4) shows that performing a leave-one-out classification of a training example is just as simple and fast as performing a classification of a test example.

3 Calculation of Expected Output Scores

When using the classification scheme given by (1) the intention is naturally to find the class that in some sense is closest to the test example in question. It means that the output score of a given class preferably should be larger the more likely it is for an example to belong to the class. Due to the $\delta_{a(\overline{x}),a(\overline{y})}$ factor in (3), the output score obtained when presenting a test example to the trained n-tuple classifier depends upon the overlap between the example presented and the examples used to train the classifier. It is obvious that besides depending on the actual training examples the scores obtained also are dependent on the number and actual choice of address lines for the individual tuples. The expected output scores for the n-tuple classifier and the training set in question can be obtained by averaging (4) over the possible choices of address lines obtained from a specific selection strategy and a given number of address lines.

In the case of random selection of the set of address lines Ψ, the calculation of the expected score becomes relatively straightforward. The relative overlap between a training example and a test example is given as

$$\Delta = 1 - H(\overline{x},\overline{y}) \,/\, N, \tag{8}$$

where N is the dimension of the examples (number of data values) and $\mathrm{H}(\overline{x},\overline{y})$ denotes the Hamming distance between the examples $\overline{x}$ and $\overline{y}$:

$$\mathrm{H}(\overline{x},\overline{y}) = \sum_{i=1}^{N}\left|x_i - y_i\right|. \tag{9}$$

With these definitions, the probability that all n inputs sampled by a LUT lie within this overlapping region is given as Δ^n. In the case with several training examples, more than one example can be responsible for a specific contribution to the output score. Accordingly, the total output score does not correspond to the sum of the individual contributions from the training examples. In calculating the expected score, it therefore becomes necessary to incorporate the effect of overlap between the training examples. From such considerations it can be found [17] that when applying an n-tuple classifier trained on specific examples, $\overline{x}_i \in \boldsymbol{X}$, on a specific test example $\overline{y}$, the expected relative output score is:

$$\begin{aligned}\left\langle \mathrm{S}_c\left(\boldsymbol{X}_c,\overline{y}\right)\right\rangle_{\Psi} = {}& \\ &\frac{1}{N^n}\sum_{i=1}^{N_x}\mathrm{A}(\overline{x}_i,\overline{y})^n - \frac{1}{N^n}\sum_{i=1}^{N_x-1}\sum_{j=i+1}^{N_x}\mathrm{A}(\overline{x}_i,\overline{x}_j,\overline{y})^n \\ &+\frac{1}{N^n}\sum_{i=1}^{N_x-2}\sum_{j=i+1}^{N_x-1}\sum_{k=j+1}^{N_x}\mathrm{A}(\overline{x}_i,\overline{x}_j,\overline{x}_k,\overline{y})^n - \ldots \\ &+(-1)^{N_x-1}\frac{1}{N^n}\mathrm{A}(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_{N_x},\overline{y})^n,\end{aligned} \tag{10}$$

$\mathrm{A}(\cdot)$ is a function that calculates the overlap defined as the number of vector co-ordinates whose values are the same for all examples in the argument list:

$$\mathrm{A}(\overline{x}_1,\overline{x}_2,\ldots,\overline{x}_m,\overline{y}) = \sum_{i=1}^{N}\prod_{j=1}^{m}\delta_{x_{ji},y_i}. \tag{11}$$

Here x_{ji} denotes the ith element of the vector $\overline{x}_j$.

Expression (10) describes the expected score that a test example obtains given a *specific* set of training examples and given that the n address lines for each tuple are picked at random. As we are actually interested in knowing how well the architecture can learn the underlying distribution of the training examples in an average sense, we should avoid this dependency on a specific training set. In order to achieve such independence, it is necessary to introduce a statistical framework for the distribution of training and test examples. Then, by averaging (10) with respect to the possible training sets of a given size, one obtains the expected score behavior for the example distributions in question.

We introduce a statistical framework by using a model, where the examples are drawn from an N-dimensional Bernoulli distribution. Let a specific class c be characterized by a prototype example $\overline{\varepsilon}_c$. Let the examples considered be represented as binary vectors of size N. The generation of an example $\overline{x}$ belonging to class c is now defined as the result of bitwise Bernoulli trials, i.e., for each bit we keep the value of the prototype with probability p_c and change the value of the bit with probability $q_c = 1 - p_c$. We assume that the training examples are obtained by a random draw from the underlying distribution. The probability of generating an example $\overline{x}$ thus is:

$$\mathrm{p}(\overline{x}) = p_c^{N-\mathrm{H}(\overline{x},\overline{\varepsilon}_c)}q_c^{\mathrm{H}(\overline{x},\overline{\varepsilon}_c)}, \tag{12}$$

where $\mathrm{H}(\overline{x},\overline{\varepsilon})$ is the Hamming distance between $\overline{x}$ and $\overline{\varepsilon}$; see (9).

As the generated examples are the outcome of N Bernoulli trials, the average Hamming distance between the generated examples and the prototype example is

$$\left\langle \mathrm{H}(\overline{x},\overline{\varepsilon}_c)\right\rangle = Nq_c. \tag{13}$$

A large average Hamming distance corresponds to a large variation or noise level (i.e., the probability, q_c, of inverting the value of a bit) for the distribution in question.

Expression (10) involves the calculation of overlap areas between different numbers of training examples and one test example. In order to average (10) over the training sets, the overlap areas must be replaced with the calculation of the average overlap areas between different numbers of

training examples and a test example. Accordingly, the new formula will involve terms of the type

$$\frac{1}{N^n}\left\langle \mathrm{Z}^n(c_1, c_2, m)\right\rangle, \tag{14}$$

where

$$\mathrm{Z}(c_1, c_2, m) = \mathrm{A}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_m, \overline{y}) \tag{15}$$

denotes a stochastic variable describing the overlap area between m examples, $\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_m$, drawn from the example distribution corresponding to a class c_1 and one test example $\overline{y}$ drawn from the example distribution corresponding to a class c_2.

Let us first consider the situation where the test example is drawn from the same distribution as the training examples, i.e., $c_2 = c_1$. The stochastic process associated with $\mathrm{Z}(c_1, c_2, m)$ is then described as N Bernoulli trials with the probability of success, p_z, given by the expression

$$\mathrm{p}_z(c_1, c_1, m) = p_{c_1}^{m+1} + q_{c_1}^{m+1}, \tag{16}$$

corresponding to the probability that one specific bit has the same value in all $m+1$ examples. Calculating $\left\langle \mathrm{Z}^n(c_1, c_1, m)\right\rangle$ thus corresponds to calculating the nth moment $\mu_n(N, \mathrm{p}_z)$ of a binomial distribution characterized by N and p_z. As all training examples are considered to originate from the same population, all area-of-overlap terms in (10) involving the same number of examples have the same average value. Accordingly, the following result is obtained for the expected output score number when the test and training examples originate from the same distribution:

$$\begin{aligned}\left\langle \mathrm{S}_{c_1}\left(\boldsymbol{X}_{c_1}, \overline{y}\right)\right\rangle_{\Psi, \boldsymbol{X}_{c_1}} = \\ \frac{1}{N^n}\binom{N_X}{1}\mu_n\left(N, \mathrm{p}_z(c_1, c_1, 1)\right) - \\ \frac{1}{N^n}\binom{N_X}{2}\mu_n\left(N, \mathrm{p}_z(c_1, c_1, 2)\right) + \\ \ldots + (-1)^{N_X+1}\frac{1}{N^n}\binom{N_X}{N_X}\mu_n\left(N, \mathrm{p}_z(c_1, c_1, N_X)\right) = \\ \frac{1}{N^n}\sum_{i=1}^{N_X}(-1)^{i+1}\binom{N_X}{i}\mu_n\left(N, \mathrm{p}_z(c_1, c_1, i)\right).\end{aligned} \tag{17}$$

The moments for the binomial distribution $\mathrm{B}(N, p)$ can be found making use of its moment generating function; see Appendix A.

The average Hamming distance between two examples belonging to the class c_1 can be calculated using (16) with $m = 1$ and the expression for the first order moment of a Binomial distribution with N trials:

$$\left\langle \mathrm{H}_{c_1, c_1}(\overline{x}_i, \overline{x}_j)\right\rangle = N\left(1 - p_{c_1}^2 - q_{c_1}^2\right), \quad i \neq j. \tag{18}$$

If the test example is drawn from a distribution different from that of the training examples, i.e., $c_2 \neq c_1$, there are two prototype examples to consider, $\overline{\varepsilon}_{c_1}$ and $\overline{\varepsilon}_{c_2}$. Let the associated probabilities of generating a bit, whose value is the inverse of the corresponding prototype bit, be denoted q_{c_1} and q_{c_2}. For the following analysis, it is helpful to divide the data of the two binary prototype examples into two sets; one set $\mathbf{S}_1$ containing all bits having the same value in the two prototypes, and another set $\mathbf{S}_2$ containing all bits of opposite values. Let the sizes of these two sets respectively be denoted N_1 and N_2, where $N_1 + N_2 = N$. In analogy with the above definition of $\mathrm{Z}(c_1, c_2, m)$, we define two stochastic variables $\mathrm{Z}_1(c_1, c_2, m)$ and $\mathrm{Z}_2(c_1, c_2, m)$ as describing the overlap area between m examples drawn from the example distribution corresponding to class c_1 and one test example drawn from the example distribution corresponding to class c_2, respectively within the set $\mathbf{S}_1$ and $\mathbf{S}_2$. The overlap areas within these two sets are denoted $\mathrm{A}_{\mathbf{S}_1}$ and $\mathrm{A}_{\mathbf{S}_2}$. With these definitions we have:

$$\mathrm{Z}_1(c_1, c_2, m) = \mathrm{A}_{\mathbf{S}_1}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_m, \overline{y}) = \sum_{i \in \mathbf{S}_1}\prod_{j=1}^{m}\delta_{x_{ji}, y_i}, \tag{19}$$

$$\mathrm{Z}_2(c_1, c_2, m) = \mathrm{A}_{\mathbf{S}_2}(\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_m, \overline{y}) = \sum_{i \in \mathbf{S}_2}\prod_{j=1}^{m}\delta_{x_{ji}, y_i}, \tag{20}$$

and

$$\mathrm{Z}(c_1, c_2, m) = \mathrm{Z}_1(c_1, c_2, m) + \mathrm{Z}_2(c_1, c_2, m). \tag{21}$$

The stochastic process associated with $\mathrm{Z}_1(c_1, c_2, m)$ is then described as N_1 Bernoulli trials with the probability of success, p_{z_1}, given by the expression

$$\mathrm{p}_{z_1}(c_1, c_2, m) = p_{c_1}^m p_{c_2} + q_{c_1}^m q_{c_2}. \tag{22}$$

The first term on the right hand side corresponds to the probability that a specific bit (from $\mathbf{S}_1$) within the m training examples has the same value as its prototype *and* that the same bit within the test example has the same value as its prototype. The second term corresponds to the probability that the specific bit within the m training examples has the opposite value as its prototype *and* that the same bit within the test example has the opposite value as its prototype. Analogously, we get for $\mathrm{Z}_2(c_1, c_2, m)$:

$$\mathrm{p}_{z_2}(c_1, c_2, m) = p_{c_1}^m q_{c_2} + q_{c_1}^m p_{c_2}. \tag{23}$$

Accordingly, the average probability that one specific bit has the same value in all $m+1$ examples is given as:

$$\begin{aligned}\mathrm{p}_z(c_1, c_2, m) = \\ \frac{1}{N}\left(\mathrm{p}_{z_1}(c_1, c_2, m)N_1 + \mathrm{p}_{z_2}(c_1, c_2, m)N_2\right).\end{aligned} \tag{24}$$

Equation (24) simplifies to (16) when $N_1 = N$ (i.e., $N_2 = 0$) and $p_{c_1} = p_{c_2}$. It is noted that the stochastic process associated with $\mathrm{Z}(c_1, c_2, m)$ is *not* equivalent to N Bernoulli trials with the probability of success, p_z, given by (24).

In analogy with (18), we can calculate the average Hamming distance between two examples belonging to class c_1 and c_2, respectively:

$$\begin{aligned}\left\langle \mathrm{H}_{c_1, c_2}(\overline{x}_i, \overline{y}_j)\right\rangle = \\ N_1\left(1 - p_{c_1}p_{c_2} - q_{c_1}q_{c_2}\right) + N_2\left(1 - q_{c_1}p_{c_2} - p_{c_1}q_{c_2}\right).\end{aligned} \tag{25}$$

Using (19)-(23), (17) can now be extended to the situation where the test example is drawn from another binomial distribution than the one used to generate the training examples:

$$\begin{aligned}
&\left\langle \mathrm{S}_{c_1}\left(\boldsymbol{X}_{c_1}, \overline{y}\right)\right\rangle_{\Psi, \boldsymbol{X}_{c_1}} \\
&= \sum_{i=1}^{N_X} (-1)^{i+1} \frac{1}{N^n} \binom{N_X}{i} \left\langle \left(\mathrm{Z}_1(c_1, c_2, i) + \mathrm{Z}_2(c_1, c_2, i)\right)^n \right\rangle \\
&= \sum_{i=1}^{N_X} (-1)^{i+1} \frac{1}{N^n} \binom{N_X}{i} \sum_{j=0}^{n} \binom{n}{j} \left\langle \mathrm{Z}_1^j(c_1, c_2, i) \right\rangle \left\langle \mathrm{Z}_2^{n-j}(c_1, c_2, i) \right\rangle \\
&= \sum_{i=1}^{N_X} (-1)^{i+1} \frac{1}{N^n} \binom{N_X}{i} \sum_{j=0}^{n} \binom{n}{j} \mu_j\left(N_1, \mathrm{p}_{z_1}(c_1, c_2, i)\right) \mu_{n-j}\left(N_2, \mathrm{p}_{z_2}(c_1, c_2, i)\right).
\end{aligned} \tag{26}$$

By letting $\overline{\varepsilon}_{c_2} = \overline{y}$ and $q_{c_2} = 0$, (26) can be used to calculate the expected output score from class c_1 on *any* test example $\overline{y}$.

4 Behavior of Output Scores

With the formulas obtained in the preceding section, we can now analyze how the expected scores for a set of n-tuples relates to the Hamming distance between training and test examples. Initially we will consider the case with only one training class.

4.1 Relationship between Output Scores and Hamming Distance

In Fig. 1 is shown a plot of the expected score $\left\langle \mathrm{S}_{c_1}\left(\boldsymbol{X}_{c_1}, \overline{y}\right)\right\rangle_{\Psi, \boldsymbol{X}_{c_1}}$ obtained as a function of the Hamming distance between the prototype $\overline{\varepsilon}_{c_1}$ of the training examples in class c_1 and a test example $\overline{y}$ for a varying number of training examples. The curves are obtained by setting $N_2 = \mathrm{H}(\overline{\varepsilon}_{c_1}, \overline{y})$ and $N_1 = N - N_2$ in (26) and by letting $\overline{\varepsilon}_{c_2} = \overline{y}$ and $q_{c_2} = 0$. Fig. 2 shows a corresponding plot but for various numbers of address lines.

The graphs in Fig. 1 illustrate that the expected score level increases with the number of training examples. The reason is that it becomes more likely that the features of the test example identified by the n-tuples also are found in a training example. Lowering the number of address lines increases the generalization ability of the classifier as it decreases the number of possible features within an n-tuple, as shown in Fig. 2. On the other hand lowering the number of address lines will in general decrease the discrimination ability of the classifier.

Another way to depict the behavior of the n-tuple net is to plot the expected score as a function of the number of address lines and the number of training examples for fixed values of the Hamming distance $\mathrm{H}(\overline{\varepsilon}_{c_1}, \overline{y})$. These relationships are shown in Figs. 3 and 4.

4.2 Likelihood and Decision Criteria

We now turn to the case with two training classes. The anticipated behavior of the n-tuple classification system is that it outputs a higher average score on class 1 than class 2 when an example actually belongs to the class 1 distribution and

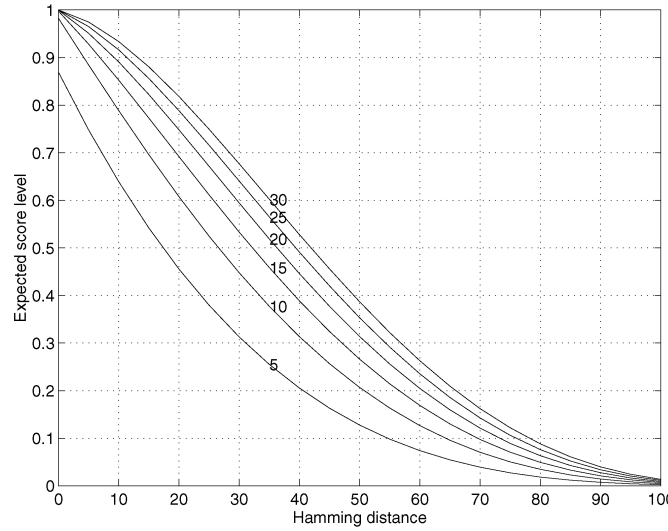

Fig. 1. Expected score as function of the Hamming distance between the prototype $\overline{\varepsilon}_{c_1}$ and the test example $\overline{y}$ shown for distinct numbers of training examples (as indicated on the curves). The number of address lines n is 5 and the example size N is 100. The noise level q_{c_1} is 20 percent.

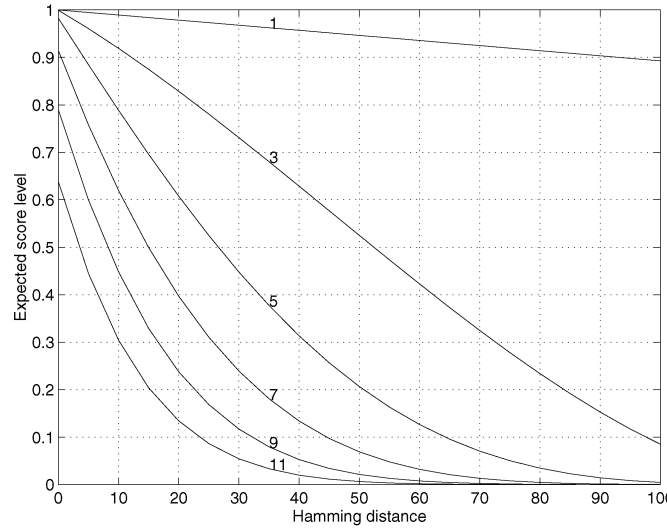

Fig. 2. Expected score as function of Hamming distance between the prototype and the test example for distinct numbers of address lines (as indicated). The number of training examples N_X is 10. Other parameters are as in Fig. 1.

vice versa. In the following analysis, we show that this in general is not to be expected.

A statistically sound decision criterion is obtained by applying the maximum a posteriori (MAP) or maximum likelihood (ML) principle. According to the MAP principle, the decision boundary is given by the examples that are equally likely to belong to either of the two distributions given the observed data; i.e., identifying these examples and calculating their expected score levels reveals the decision boundary in the score space. If the two classes are assumed equally likely, this becomes the ML decision boundary.

By using (12) and assuming equal class likelihood $\mathrm{p}(c_1) = \mathrm{p}(c_2)$, we can initially relate the Hamming distances between a test example and two training prototypes of different classes to the conditional probabilities

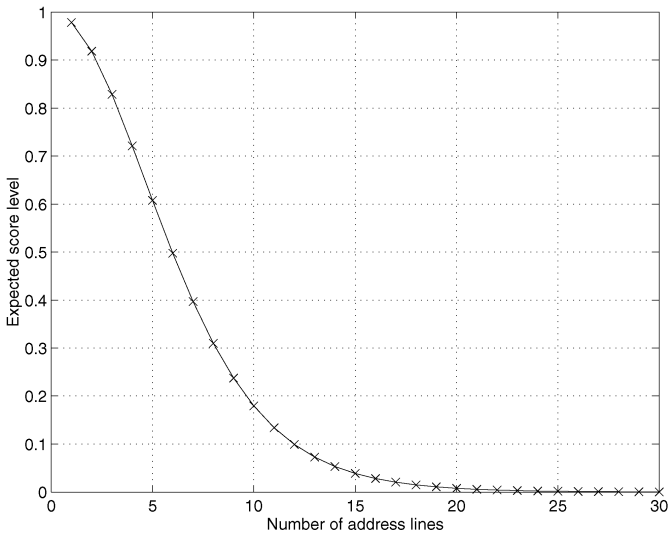

Fig. 3. Expected score as function of the number of address lines for an n-tuple classifier trained on 10 examples. Hamming distance between the prototype and the test example is 20. The noise level q_{c_1} is 20 percent and the example size N is 100.

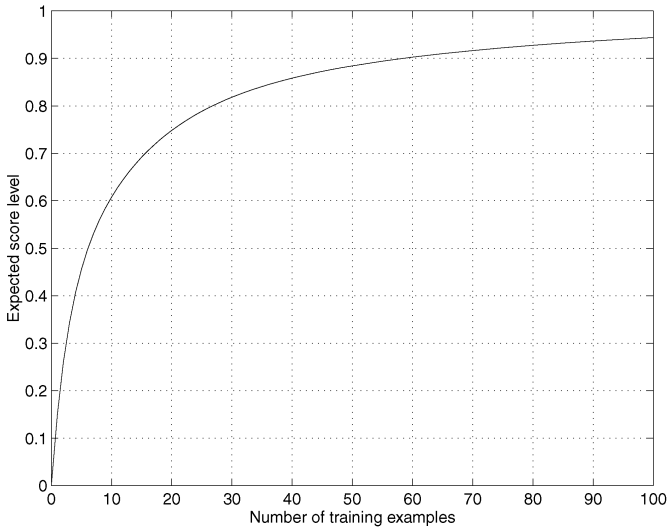

Fig. 4. Expected score as function of the number of training examples. Five address lines were used per n-tuple and the other parameters are as in Fig. 3.

$\mathrm{p}(c_1|\overline{y})$ and $\mathrm{p}(c_2|\overline{y})$; i.e., the posterior probabilities that example $\overline{y}$ is generated from class c_1 and c_2, respectively. Fig. 5 shows how this relation splits the "Hamming space" into two regions; one region where class 1 is the most likely and vice versa. It is important to note that *only* if the stochastic parameters p_{c_1} and p_{c_2} of the two training classes are equal is the decision boundary in Hamming space given by the straight line $\mathrm{H}(\overline{y},\overline{\varepsilon}_1) = \mathrm{H}(\overline{y},\overline{\varepsilon}_2)$. For brevity, we have used $\overline{\varepsilon}_1$ for $\overline{\varepsilon}_{c_1}$ and $\overline{\varepsilon}_2$ for $\overline{\varepsilon}_{c_2}$.

In order to study the relationship between Hamming distances and conditional probabilities in more detail we consider the test examples obtained by moving from one prototype to the other in such a way that the sum of the two Hamming distances is kept constant. This corresponds to moving along the dotted line shown in Fig. 5. Along this

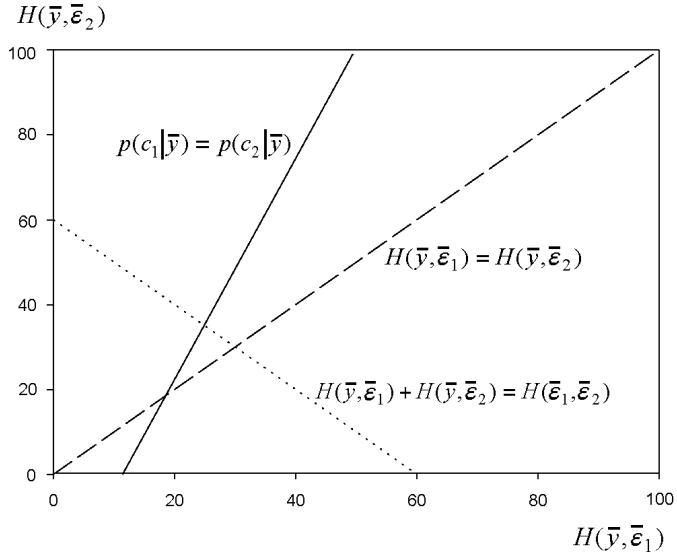

Fig. 5. Decision borders in Hamming space corresponding to the MAP-principle. Only if the stochastic parameters p_{c_1} and p_{c_2} are equal is the decision border given as $\mathrm{H}(\overline{y},\overline{\varepsilon}_1) = \mathrm{H}(\overline{y},\overline{\varepsilon}_2)$. The decision border corresponding to the case where $q_{c_1} = 0.1$ and $q_{c_2} = 0.3$ is shown (solid line). The data in Figs. 6-10 are calculated along lines parallel to the dotted line; here the line is shown for the case with $H(\overline{\varepsilon}_1,\overline{\varepsilon}_2) = 60$.

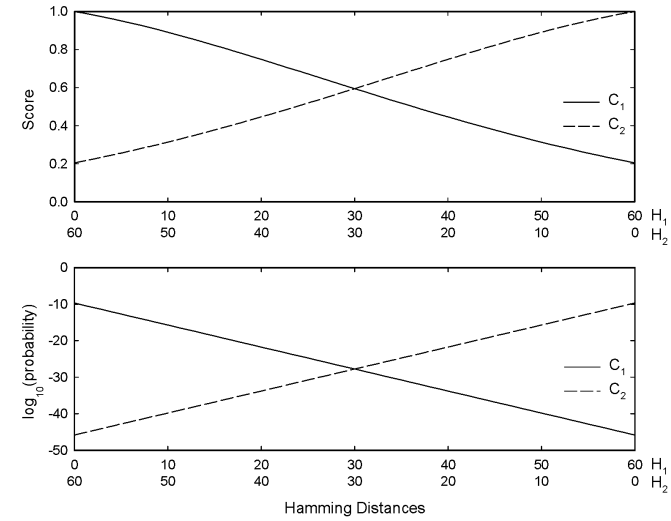

Fig. 6. Expected scores and conditional probability as function of the Hamming distance along the dotted line in Fig. 5. The Hamming distance between the prototypes is $H(\overline{\varepsilon}_1,\overline{\varepsilon}_2) = 60$. The noise levels are $q_1 = q_2 = 0.2$. Number of address lines $n = 5$. Example size $N = 100$. Numbers of examples in each of the two training classes: $N_{X_1} = N_{X_2} = 20$. $H_1 = H(\overline{y},\overline{\varepsilon}_1)$ and $H_2 = H(\overline{y},\overline{\varepsilon}_2)$.

line, we can calculate the conditional probabilities as well as the expected output scores by using (12) and (26) twice. The resulting curves are depicted in Figs. 6-10 for different characteristics of the training sets and example distributions. As seen in Figs. 6 and 7, increasing the distance between the prototypes gives better discrimination/separation between the score curves. In Figs. 8-10 it is important to note that different noise levels, different population sizes, and a difference in the numbers of address

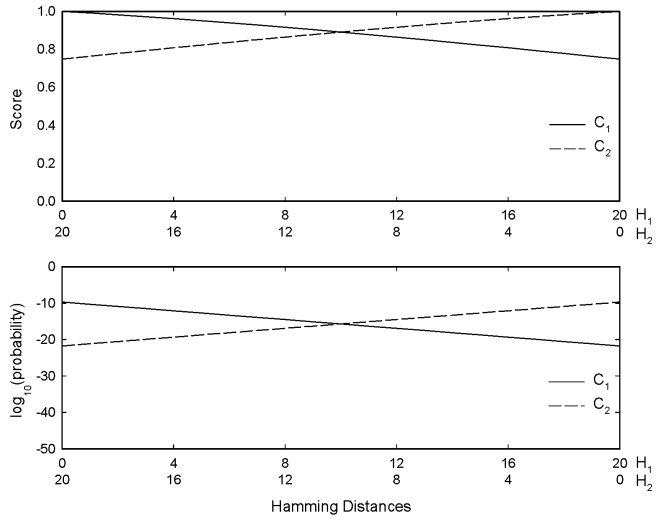

Fig. 7. As Fig. 6 but with $H(\bar{\varepsilon}_1, \bar{\varepsilon}_2) = 20$.

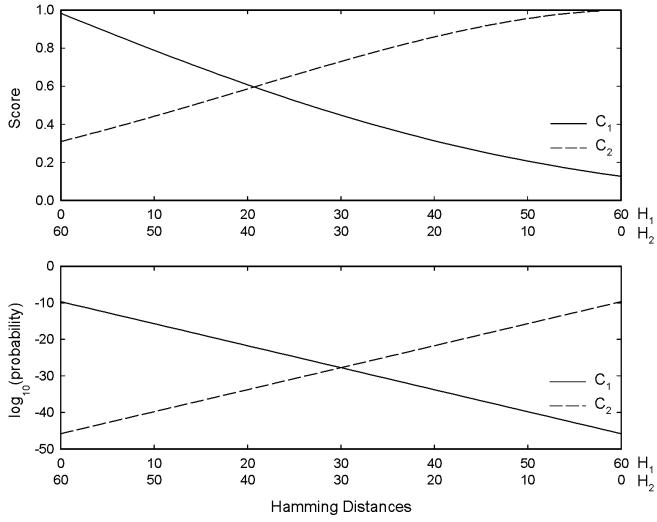

Fig. 8. As Fig. 6 but with $N_{x_1} = 10$ and $N_{x_2} = 40$.

lines used for the two classes cause the score curves to cross at a Hamming distance different from where the conditional probability curves cross each other. Consequently, equal score levels for two classes do no necessarily correspond to equal likelihood. This fact is actually the key to understanding the problems with the standard n-tuple architecture, which often performs rather poorly when the training sets have skewed priors. However, by considering this behavior it becomes possible to avoid this limitation of the n-tuple system. More specifically, the decision boundary in score space should be made to depend on the spatial density of the underlying example distributions, the number of training examples as well as the number of address lines used for the different classes.

As mentioned above, the decision boundary dictated by the maximum a posteriori principle is given by the points where $p(c_1|\bar{y})$ is equal to $p(c_2|\bar{y})$. Given a test example $\bar{y}$ with a Hamming distance H_1 to the prototype $\bar{\varepsilon}_1$, and H_2 to the prototype $\bar{\varepsilon}_2$, these probabilities are given by (12) if

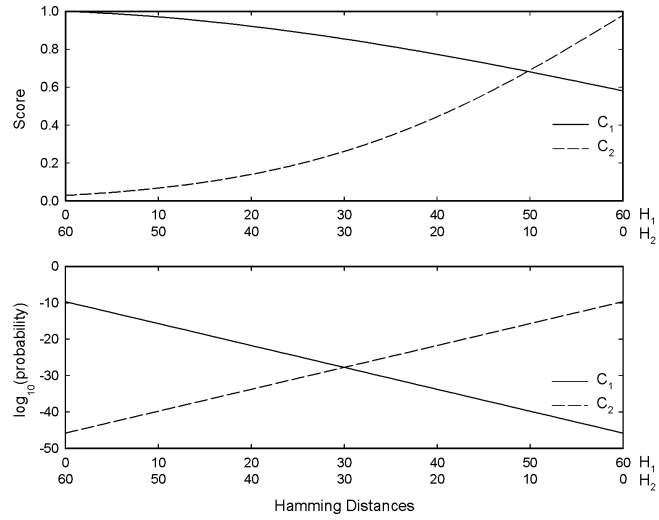

Fig. 9. As Fig. 6 but now the number of address lines is different for the two training classes. Training class c_1 uses $n = 3$ and training class c_2 uses $n = 8$.

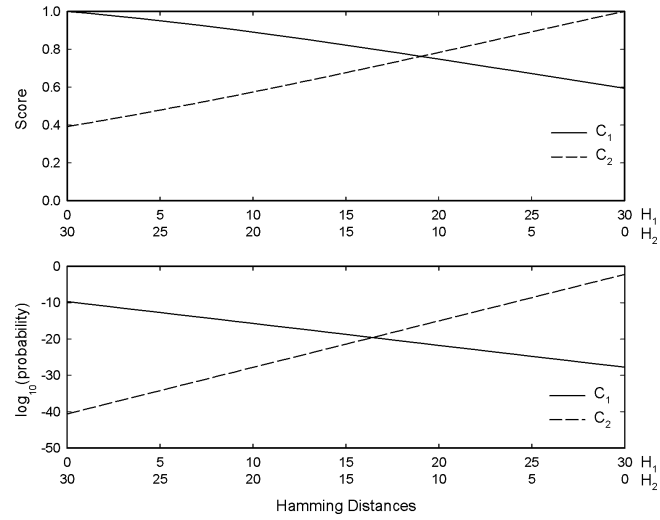

Fig. 10. Expected scores and conditional probability as function of the Hamming distance along the line equivalent to the dotted path in Fig. 5. The Hamming distance between the prototypes is $H(\bar{\varepsilon}_1, \bar{\varepsilon}_2) = 30$. The noise levels are $q_1 = 0.20$ and $q_2 = 0.05$. The number of address lines n is 5. Example size $N = 100$. Numbers of examples in each of the two training classes: $N_{x_1} = N_{x_2} = 20$. $H_1 = H(\bar{y}, \bar{\varepsilon}_1)$ and $H_2 = H(\bar{y}, \bar{\varepsilon}_2)$.

we assume equal class likelihood. This condition as well as the demand that H_1 and H_2 must take on values between 0 and N defines the decision boundary in Hamming space:

$$H_2 = \frac{\ln\left(p_{c_1}^{N-H_1} q_{c_1}^{H_1} p_{c_2}^{-N}\right)}{\ln\left(q_{c_2} / p_{c_2}\right)},$$

$$H_1 \in \left[\max\left(0, \frac{N\left(\ln(p_{c_2} / p_{c_1})\right)}{\ln(q_{c_1} / p_{c_1})}\right), \min\left(N, \frac{N\left(\ln(q_{c_2} / p_{c_1})\right)}{\ln(q_{c_1} / p_{c_1})}\right)\right],$$

$$p_{c_2} \neq \frac{1}{2}, p_{c_1} \neq \frac{1}{2} \tag{27}$$

(if $p_{c_1} = 1/2$ or $p_{c_2} = 1/2$ the corresponding class corresponds to one where all possible patterns are equally likely,

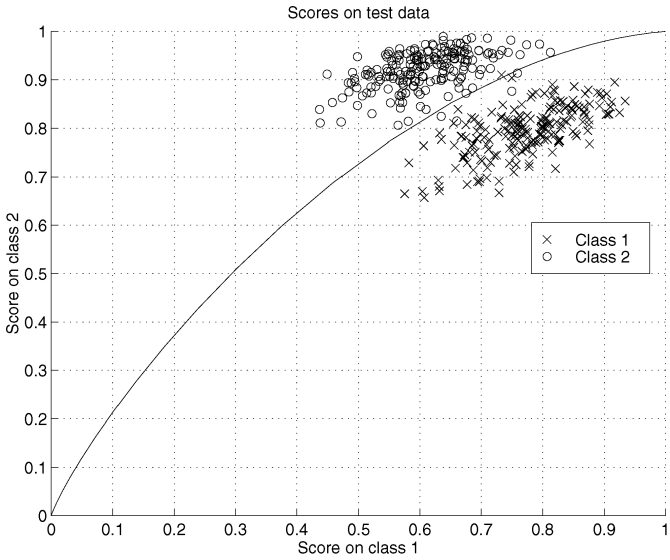

Fig. 11. Decision border according to the MAP principle. $q_1 = q_2 = 0.1$, $N_{x_1} = 10$, $N_{x_2} = 40$, $N = 100$, $n = 5$. Error rates on depicted test data using WTA is 30.3 percent and using the MAP border, it is 1.3 percent.

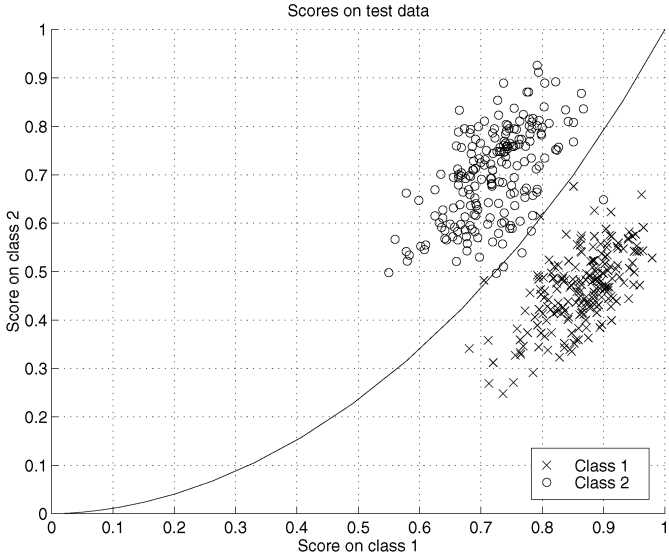

Fig. 12. Decision border according to the MAP principle. $q_1 = q_2 = 0.1$, $N_{x_1} = N_{x_2} = 20$, $N = 100$, $n = 5$ and $n = 8$ for class c_1 and c_2, respectively. Error rates on depicted test data using WTA is 27.3 percent and using the MAP border it is 1.8 percent.

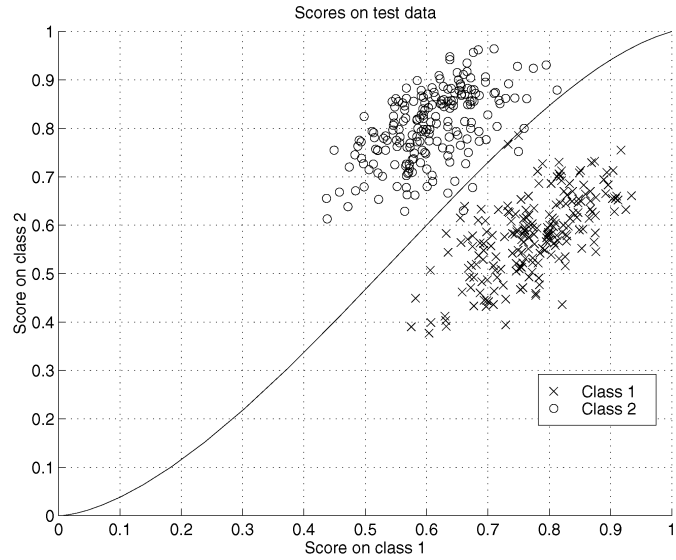

Fig. 13. Decision border according to the MAP principle. $q_1 = q_2 = 0.1$, $N_{x_1} = 10$, $N_{x_2} = 40$, $N = 100$, $n = 5$ and $n = 8$ for class c_1 and c_2, respectively. Error rates on depicted test data using WTA is 0.8 percent and using the MAP border it is 1.0 percent.

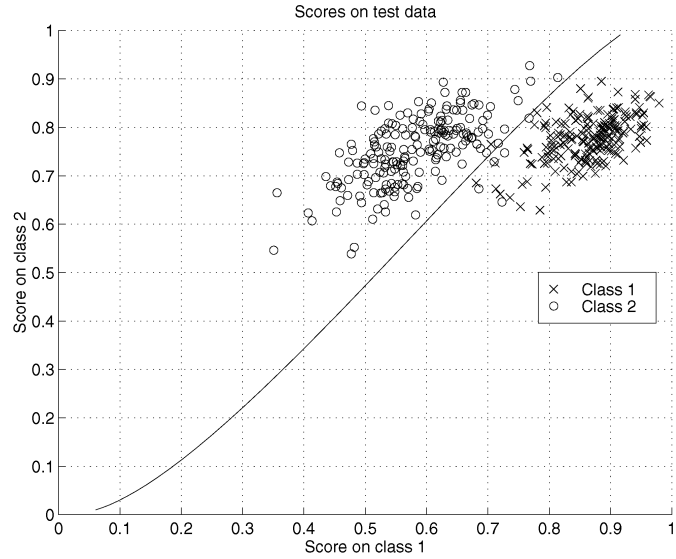

Fig. 14. Decision border according to the MAP principle. $q_1 = 0.1$, $q_2 = 0.2$, $N_{x_1} = N_{x_2} = 20$, $N = 100$, $n = 5$. Error rates on depicted test data using WTA is 2.5 percent and using the MAP border it is 1.5 percent.

independent of the prototype.) It should be noted that for any specific choice of prototypes in will in general not be possible to construct examples that covers the whole range of variation.

With the MAP-decision border in Hamming space given by (27), (26) is now used to obtain the corresponding expected score values. Examples of decision borders in score space are shown in Figs. 11-15. The depicted clusters of data points are obtained from Monte Carlo simulations with 200 test examples drawn from each of the two example distribution. The corresponding scores are average values over a number of n-tuple classifiers trained on separate training sets, which were drawn at random from the underlying example distributions. The two training prototypes are separated by a Hamming distance of 10. Larger Hamming distances between the prototypes will give better separation between the two clusters and vice versa. The error rates obtained using respectively the traditional WTA principle and the new decision borders are given in the figures.

The data clusters in Figs. 11-15 illustrate clearly that the conventional decision line given as

$$\left\langle \mathrm{S}_{c_1}\left(\boldsymbol{X}_{c_1}, \bar{y}\right)\right\rangle_{\Psi, \boldsymbol{X}_{c_1}} = \left\langle \mathrm{S}_{c_2}\left(\boldsymbol{X}_{c_2}, \bar{y}\right)\right\rangle_{\Psi, \boldsymbol{X}_{c_2}}$$

will, in general, not be applicable unless the Hamming distance between the prototypes is sufficiently large. Indeed,

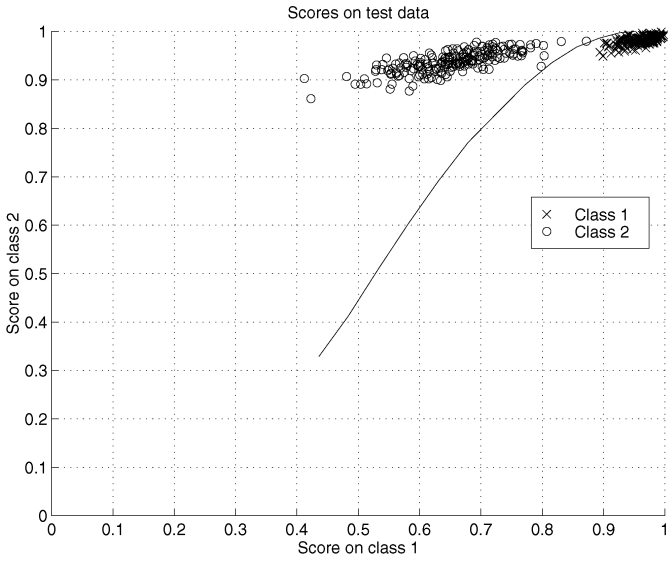

Fig. 15. Decision border according to the MAP principle. $q_1 = 0.1$, $q_2 = 0.3$, $N_{X_1} = N_{X_2} = 100$, $N = 100$, $n = 5$. Error rates on depicted test data using WTA is 43.5 percent and using the MAP border it is 0 percent.

the decision lines based on MAP considerations separates the data clusters in a desired manner.

The pure effect of having different number of examples in competing training classes is illustrated in Fig. 11. The class represented with most training examples will on average be better described than the smaller one. This in general implies that for a given Hamming distance to the training prototypes the likelihood of obtaining votes on the better-represented class is increased as is also illustrated in Fig. 8. By applying the MAP principle the different score levels are taken in to account and on the depicted data set the test error rate decreases from 30.3 percent to 1.3 percent.

The number of address lines used in the n-tuples also affects the score levels. When the number of address lines is increased, the expected output score will decrease due to a more sparse coverage of the addressable columns. This behavior is depicted in Fig. 9. When two training classes are sampled with a different number of address lines, this behavior affects the MAP-decision border as shown in Fig. 12. Again, the border has shifted from the standard n-tuple decision line to counteract the different levels of the scores.

As shown in Fig. 13, a difference in the number of address lines can counteract the effect caused by different numbers of training examples in the involved classes. The test error rates obtained on the depicted test examples using the WTA scheme and the MAP scheme are almost identical in this case. Therefore, if the WTA-principle was kept skewed priors could be dealt with by choosing the number of address lines individually for each class. In general, however, restricting the number of address lines unnecessarily might not be advisable, since this parameter can be used to improve the overall performance [11].

Another important characteristic of the training set, which influences the position of the MAP decision curve, is the noise level of the distribution. For the underlying example distributions used in the above analysis, the noise level constitutes the variation of the class "around" its prototype. If the noise level (i.e., the example variation) is not the same in the different classes, the MAP-decision border will deviate from the WTA-border as shown in Figs. 14-15. Fig. 15 illustrates a case where the characteristics of the training sets and the parameters of the n-tuple classifier make the WTA-principle unusable. On the other hand, the MAP-decision border can separate the classes without errors on the test set.

5 Implications of Analysis on Real Data

In Section 4, we saw that at least for one statistical framework it is possible to modify the n-tuple classifier so that it on average operates as the Bayesian or maximum likelihood classifier. It is straightforward to incorporate general costs associated with different decisions into the scheme. However, in order to calculate the appropriate decision borders we made use of the stochastic parameters characterising the underlying distributions. In real world problems, the training examples are usually drawn from an unknown distribution and in the few cases where sufficient knowledge about the underlying distribution exists, one would often be better off using this information directly to derive the MAP or ML estimator.

In the case with unknown or insufficient information about the underlying example distributions, it is impossible to determine the decision border from calculations as those shown in Section 4. Still, however, the theoretical analysis tells us how the decision borders would actually be for different scenarios if we knew them. This knowledge is useful for determining the variability we should allow when determining the new decision rules. By combining this variability with a suitable estimate of the test error, it would actually become possible to determine feasible decision borders for real cases. The approach would be to choose from the set of possible borders the one that minimised the estimated test error. A good estimate of the test error (and test output scores) can be obtained using a leave-one-out cross-validation test on the training examples. As shown in eq. (6) it is simple to perform such a test on an n-tuple architecture. Accordingly, it becomes also simple to utilise the implications of the analysis in Section 4. The way this is achieved is to construct a decision border that minimises the number of obtained cross-validation errors. It is outside the scope of this paper to address in detail the possible schemes for determining the decision borders as well as the related computational overhead. These topics will be addressed elsewhere. However, one possible scheme that can produce a family of curves that qualitatively look similar to those depicted in Figs. 11-15 is to use a Bézier curve with four control points. The first and last control points are positioned in (0, 0) and (1, 1) of a given two-dimensional score plot and the two remaining control points are chosen so that the resulting decision border minimises the leave-one-out cross-validation error.

The so-called BelgianII data set [19] is one of the data sets that have been used for comparative trials in the StatLog project [13]. Results obtained by applying a standard n-tuple classifier on this data set has also been reported [16]. The n-tuple results reported are rather poor when compared

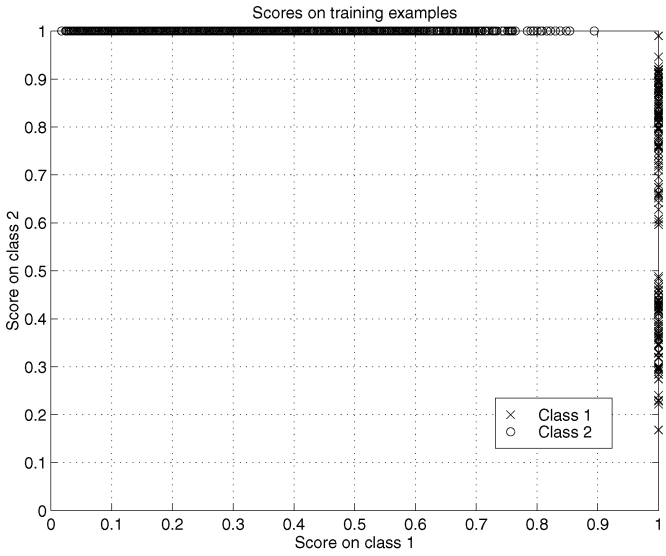

Fig. 16. Scores on training examples for the BelgianII data set. There are 2,000 training examples each consisting of 57 real attributes, which are thermometer coded using 20 bits.

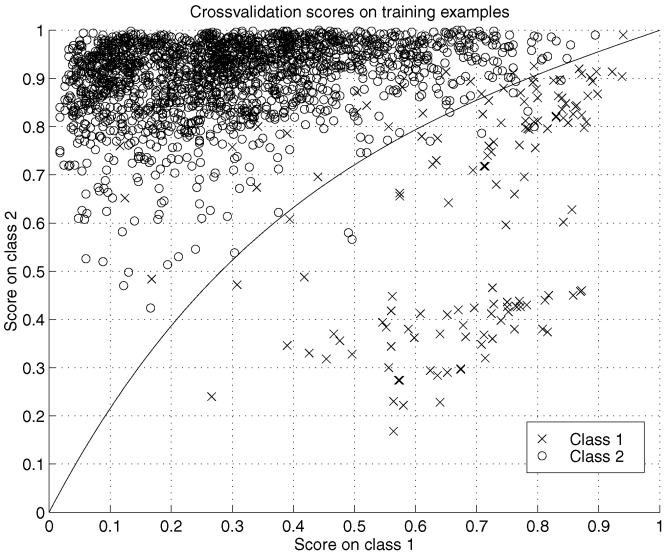

Fig. 17. Cross-validation scores for the BelgianII training set. Estimated decision border is shown. Cross-validation error rate with the new decision border is 1.9 percent and with the WTA principle it is 3.5 percent.

with the other methods that were tested in the StatLog project. As one of the two training classes dominates the training set (as well as the test set), we have a situation with skewed priors. From the above analysis, it can therefore be expected to improve the results by replacing the WTA principle with a new decision border. We performed training using 500 n-tuples (or LUTs) each with n equal to 20. Half of the address lines were selected at random, and the other half was guided by a principal component analysis of the input. The scores obtained on the training examples are depicted in Fig. 16. As all training examples obtain the largest possible score on their corresponding classes, it is obvious that Fig. 16 does not provide a basis for determining the MAP decision border. Fig. 17 shows the leave-one-out cross-validation scores obtained on the training set together with a decision border minimising the cross-validation errors.

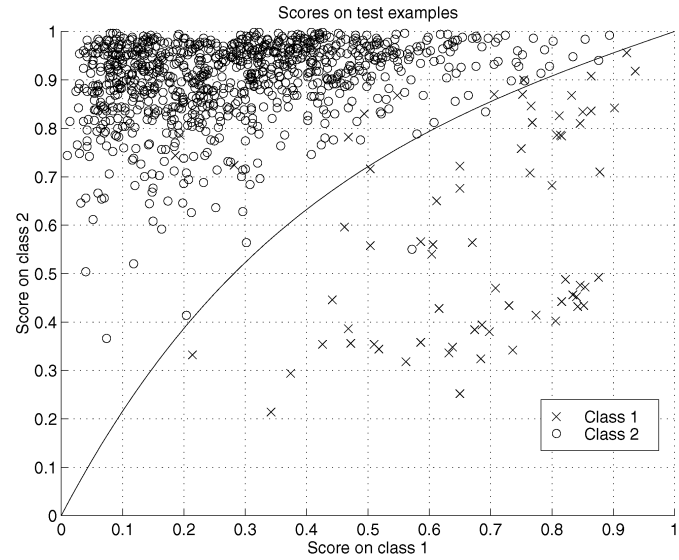

Fig. 18. Scores for the 1,000 test examples in the BelgianII data set. Decision border obtained from cross-validation scores is shown. Test error rate with the new decision border is 1.1 percent and with the WTA principle it is 2.5 percent.

The decision border is a Bézier curve determined by using the procedure given above. The implication of using this decision border on the test set is shown in Fig. 18. The resulting test error rate is 1.1 percent, which should be compared with an error rate of 2.5 percent obtained using the WTA-principle.

Fig. 19 shows individual class error rates obtained by training and testing on segmented handwritten digits from the NIST database [15]. The digits were scaled to a 16×16 binary representation. During training, we used 100 examples for each of the digit classes 0, 3, and 6, whereas 500 examples were used for each of the remaining classes. The test set consisted of 10,000 examples distributed with 1,000 in each class. The number of address lines per tuple was 10 and identical for all classes. All address lines were picked at random and the net consisted of 1,000 n-tuples. As seen from Fig. 19, use of the normal classification scheme (WTA) leads to very large error rates on the small training classes; both in a cross-validation evaluation on the training examples and in classifying the test set. From the results of the theoretical investigation in Section 4, it would be expected that proper design of the decision borders between the classes in score space can lead to an improved performance. As with the BelgianII data set we designed the new decision borders to minimise the cross-validation error rate. The implication of using the new decision criteria on the test set is also illustrated in Fig. 19. It is observed that the error rates on the classes with low representation in the training set have obtained a level close to that of the better-represented classes. The results demonstrate quite convincingly the impact of the novel decision scheme.

6 Discussion

The expected score levels calculated in Section 4 corresponds to the situation where all n-tuples used for a specific class use the same number of address lines n. It is, however, simple to

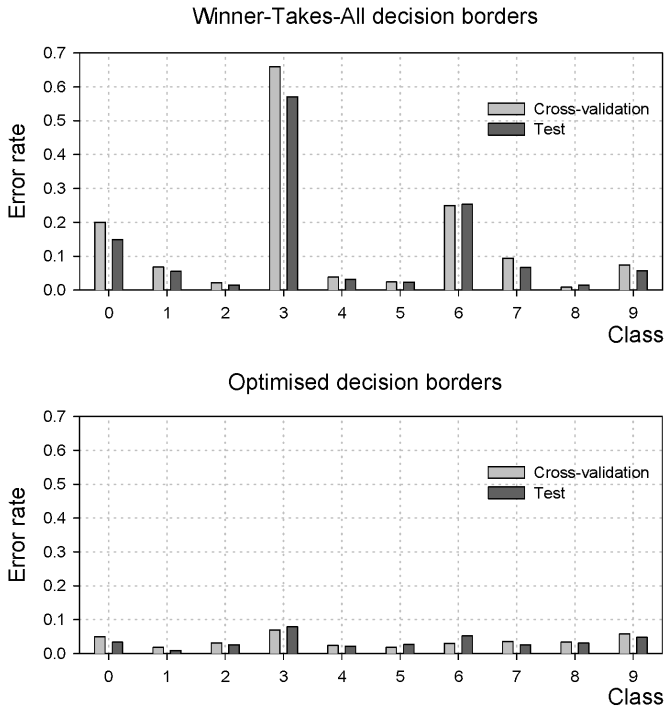

Fig. 19. Error rates obtained on segmented handwritten digits using, respectively, WTA decision borders and optimized decision borders.

extend the formulas to the cases where different numbers of address lines are used. The overall expected score is obtained by producing a weighted sum of expected scores calculated using the individually values of n. The weights simply correspond to the relative occurrences of the respective values of n.

In the above analysis, we focused on two different training classes. It is, however, not a problem to treat several classes. For all classes, the expected score levels are calculated using (26) and for each class we can by use of (12) find the conditional probabilities of the class given the data. Again, we can use a ML or MAP principle as decision criterion and then determine for each class the region in output score space where it is the most likely class to occur. For real applications, the decision regions are found by minimizing the cross-validation error rate as described in Section 5. A detailed description of possible schemes for adjusting the decision borders will be given elsewhere.

Although the statistical framework we introduced does not consider statistical correlation between different input data, it has been able to explain the essential behavior observed using n-tuple classifiers. In addition, our analysis only involves the cases where the address lines are picked completely at random although in many cases it is desirable to select the address lines using, e.g., a priori knowledge. It is clear that such modifications of our framework would lead to changes in the theoretical expressions. However, it can be expected that the overall qualitative dependency of the output scores on the number of address lines, the numbers of training examples, and the noise within the training classes, will be the same. We therefore believe that the main conclusions of our analysis and the derived implications on the decision boundary are valid for a broad range of real world data, as is also confirmed from our results given in Section 5.

As a consequence of the results given in Section 4, the classification obtained using the n-tuple net does not necessarily correspond to the class having the largest score. Accordingly the difference in score levels will in general not give much sense as a confidence level. A more reasonable measure will be the distance to the MAP or ML decision border. A more thorough analysis of the classification confidence is a subject for future research.

7 Conclusion

In spite of the simplicity and small computational needs of the n-tuple classifier, there have been many reports in the literature on competitive performances obtained with this architecture. On the other hand, there are also many examples where the standard n-tuple architecture performs very poorly when compared with most other classification schemes. Due to lack of a proper theory describing the architecture, it has been difficult to explain why and when the n-tuple classifier can be expected to perform well, although experimental studies have given some ideas.

This paper has introduced a statistical framework that has provided the possibility of performing a theoretical analysis of the connection between the output scores of the n-tuple net and the underlying distributions of the training data. The results are able to explain the experimental behavior of the classifier and shows that the normally used winner-takes-all principle in general is not applicable. We have shown that by modifying the decision criteria the classifier is likely to operate as a close approximation to the Bayes or maximum likelihood estimator. We have also devised a scheme that makes it possible to take advantage of the theoretical insight in real learning cases. One type of problem that can be handled with this novel scheme is the one caused by skewed training priors that often has caused the n-tuple classifier to perform unsatisfactorily.

Appendix A

The moments for the binomial distribution $B(N, p)$ can be found making use of its moment generating function; see e.g., [18],

$$\mathrm{M}(t) = \sum_{l=0}^{N} \binom{N}{l} \left(p \exp(t)\right)^{l} (1-p)^{N-l},$$

and the nth moment is given by the formula:

$$\mu_n = \mathrm{M}^{(n)}(0),$$

where $\mathrm{M}^{(n)}$ denotes the nth derivative. The first four moments are listed below:

$$\begin{aligned}
\mu_0 &= 1,\\
\mu_1 &= Np,\\
\mu_2 &= Np(1-p+Np),\\
\mu_3 &= Np(1-3p+3Np+2p^2-3Np^2+N^2p^2).
\end{aligned}$$

APPENDIX B
NOMENCLATURE

$\boldsymbol{X}$: Set of examples.

$\boldsymbol{X}_c$: Set of examples belonging to class c.

$\overline{x}$: A training example.

$\overline{y}$: A test example.

$\overline{x}_j$: The jth example from the set $\boldsymbol{X}$.

x_i: ith value of example $\overline{x}$.

x_{ji}: ith value of example $\overline{x}_j$.

N_x: Number of examples in the set $\mathbf{X}$.

N: Total number of inputs to the n-tuple classifier.

n: Number of inputs to each n-tuple.

c: Class label.

$C_T(\overline{x})$: True class label corresponding to example $\overline{x}$.

Ω: Set of LUTs.

N_Ω: Number of LUTs.

$\mathrm{a}_i(\overline{y})$: Index of the column in the ith LUT being addressed by example $\overline{y}$.

Ψ: Set of address lines used in the n-tuple net.

ψ_{ij}: Index (also denoted the *address line*) of the jth input sampled by the ith LUT.

p_c: Success rate of binomial process associated with prototype $\overline{\varepsilon}_c$.

$q_c = 1 - p_c$: Noise associated with prototype $\overline{\varepsilon}_c$.

$v_{a_i(\overline{y}),c}$: The number of training examples accessing the LUT cell addressed in the ith LUT by the example $\overline{y}$ and the class c.

ACKNOWLEDGMENT

The authors thank Mr Louis Wehenkel of Université de Liége for permission to use and report results on the BelgianII data set.

REFERENCES

[1] I. Aleksander, "Microcircuit Learning Nets: Hamming-Distance Behaviour," *Electronic Letters*, vol. 6, pp. 134-136, 1970.

[2] I. Aleksander and H. Morton, *An Introduction to Neural Computing*. London: Chapman and Hall, 1990.

[3] I. Aleksander, W.V. Thomas, and P.A. Bowden, "Wisard: A Radical Step Forward in Image Recognition," *Sensor Review*, pp. 120-124, 1984.

[4] A.W. Andersen, S.S. Christensen, T.M. Jørgensen, and C. Liisberg, "An Active Vision System for Robot Guidance Using a Low Cost Neural Network Board," *Proc. Machine Vision Applications, Architectures, and Systems Integration*, Boston, Mass., pp. 163-170, 1994.

[5] J. Austin, *RAM-Based Neural Networks*. London: World Scientific, 1998.

[6] W.W. Bledsoe and I. Browning, "Pattern Recognition and Reading by Machine," *Proc. Eastern Joint Computer Conf.*, pp. 225-232, 1959.

[7] W.W Bledsoe, "Further Results on the n-tuple Pattern Recognition Method," *IRE Trans. Electronic Computers (Correspondence)*, vol. EC-10, p. 96, 1961.

[8] D. Jung, M.S. Krishnamoorthy, G. Nagy, and A. Shapira, "N-Tuple Features for OCR Revisited", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 7, pp. 734-745, July 1996.

[9] T.M. Jørgensen, "Classification of Handwritten Digits Using a RAM Neural Net Architecture," *Int'l J. Neural Systems*, vol. 8, no. 1, pp. 17-25, 1997.

[10] T.M. Jørgensen, S.S. Christensen, A.W. Andersen, and C. Liisberg, "Optimization and Application of a RAM Based Neural Network for Fast Image Processing Tasks," *Intelligent Robots and Computer Vision*, Boston, Mass., , pp. 328-338, 31 Oct-2 Nov 1994.

[11] T.M. Jørgensen, S.S. Christensen, and C. Liisberg, "Cross-Validation and Information Measures for RAM Based Neural Networks," *RAM-Based Neural Networks*, J. Austin, ed, pp. 78-88. London: World Scientific, 1998.

[12] J.V. Kennedy, J. Austin, R. Pack, and B. Cass, "C-NNAP—A Parallel Processing Architecture for Binary Neural Networks," *Proc. IEEE Int'l Conf. Neural Networks*, vol. 2, 1995, pp. 1,037-1,041.

[13] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

[14] M. Morciniec and R. Rohwer, "Good-Turing Estimation for the Frequentist n-tuple Classifier," *Proc. Weightless Neural Network Workshop 1995*, Computing with Logical Neurons, Univ. of Kent, pp. 93-98, 1995.

[15] Nat'l Inst. Standards and Technology, *NIST Special Database 19, Handprinted Forms and Characters Database*, HFCD Rel. 21-1.1, 1995.

[16] R. Rohwer and M. Morciniec, "The Theoretical and Experimental Status of the n-tuple Classifier", *Neural Networks*, vol. 11, no. 1, pp. 1-14, 1998.

[17] T.J. Stonham, "Improved Hamming-Distance Analysis for Digital Learning Networks," *Electronics Letters*, vol. 13, no. 6, pp. 155-156, 1977.

[18] D.D. Wackerly, I. William, R.L. Mendenhall, and L. Scheaffer, *Mathematical Statistics with Applications*. Belmont: Duxbury Press, June 1996.

[19] L. Wehenkel, M. Pavella, E. Euxibie, and B. Heilbronn, "Decision Tree Based Transient Stability Assessment—A Case Study," *Proc. IEEE/PES 1993 Winter Meeting*, Columbus, Ohio, 1993.

Thomas Martini Jørgensen received the Master's degree and PhD degree from the Technical University of Denmark in 1989 and 1992, respectively. Since 1992, he has been employed at Risø National Laboratory, now as a senior scientist, with a research effort focused on application of image processing in combination with neural networks, especially the n-tuple classifier.

Christian Linneberg received the Master's degree from the Technical University of Denmark in 1994. He joined the Risø National Laboratory as a scientist in 1996. Since 1998, he has been an employee of Intellix, where he is pursuing a PhD degree in the area of machine learning with focus on the n-tuple classifier.