(12) PATENT (11) Application No. AU 199944986 B2
(19) AUSTRALIAN PATENT OFFICE (10) Patent No. 753822

(54) Title
N-tuple or ram based neural network classification system and method

(51)[7] International Patent Classification(s)
G06F 015/80

(21) Application No: **199944986** (22) Application Date: **1999.06.21**

(87) WIPO No: **WO99/67694**

(30) Priority Data

(31) Number	(32) Date	(33) Country
PA 1998 00883	**1998.06.23**	**DK**

(43) Publication Date : **2000.01.10**
(43) Publication Journal Date : **2000.03.23**
(44) Accepted Journal Date : **2002.10.31**

(71) Applicant(s)
Intellix A/S

(72) Inventor(s)
Christian Linneberg; Thomas Martini Jorgensen

(74) Agent/Attorney
BALDWIN SHELSTON WATERS,Level 21,60 Margaret Street,SYDNEY NSW 2000

(56) Related Art
JORGENSSEN,T.M. INT.JOUR.NEUR.SYS. V8,NO.8, FEB 1997 P.17-25
JORGENSSEN ET AL. TRANSACTIONS ON PATTERN ANALYSIS MACHINE INTELLIGENC, V.10, NO.4 APRIL 1999

44986/99

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : **G06F**	**A2**	**(11) International Publication Number:** **WO 99/67694** **(43) International Publication Date:** 29 December 1999 (29.12.99)

(21) International Application Number: PCT/DK99/00340

(22) International Filing Date: 21 June 1999 (21.06.99)

(30) Priority Data:
PA 1998 00883 23 June 1998 (23.06.98) DK

~~**(71) Applicant** *(for all designated States except US)*: RISØ [DK/DK]; Frederiksborgvej 399, DK-4000 Roskilde (DK).~~

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: LINNEBERG, Christian [DK/DK]; Ægirsgade 56, 4.th, DK-2200 Copenhagen N (DK). JØRGENSEN, Thomas, Martini [DK/DK]; Kildegårdsvej 29, DK-3650 Ølstykke (DK).

(74) Agent: HØIBERG APS; Nørre Farimagsgade 37, DK-1364 Copenhagen K (DK).

(71) INTELLIX A/S
H.C. Orstedsvej 4
DK-1879 Frederiksberg C
Denmark

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: N-TUPLE OR RAM BASED NEURAL NETWORK CLASSIFICATION SYSTEM AND METHOD

(57) Abstract

The invention relates to n-tuple or RAM based neural network classification methods and systems and, more particularly, to n-tuple or RAM based classification systems where the decision criteria applied to obtain the output sources and compare these output sources to obtain a classification are determined during a training process. Accordingly, the invention relates to a system and a method of training a computer classification system which can be defined by a network comprising a number of n-tuples or Look Up Tables (LUTs), with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of possible classes and comprising columns being addressed by signals or elements of sampled training input data examples.

1. Field of the Invention

The present invention relates to a method and a system for training a computer classification system, which is defined by a network comprising a number of n-tuple or Look Up Tables (LUTs).

The invention has been developed primarily for n-tuple or RAM based neural network classification systems where the decision criteria is applied to obtain the output scores and to compare these output scores so as to obtain a classification as determined during a training process. While the invention will be described with reference to this application, it is not limited to that particular field of use.

2. Description of the Prior Art

A known way of classifying objects or patterns represented by electric signals or binary codes and, more precisely, by vectors of signals applied to the inputs of neural network classification systems which lie in the implementation of a so-called learning or training phase. This phase generally consists of the configuration of a classification network that fulfils a function of performing the envisaged classification as efficiently as possible by using one or more sets of signals, called learning or training sets, where the membership of each of these signals in one of the classes in which it is desired to classify them is known. This method is known as supervised learning or learning with a teacher.

A subclass of classification networks using supervised learning are networks using memory-based learning. Here, one of the oldest memory-based networks is the "n-tuple network" proposed by Bledsoe and Browning (Bledsoe, W.W. and Browning, I, 1959, "Pattern recognition and reading by machine", Proceedings of the Eastern Joint Computer Conference, pp. 225-232) and more recently described by Morciniec and Rohwer (Morciniec, M. and Rohwer, R.,1996, "A theoretical and experimental account of n-tuple classifier performance", Neural Comp., pp. 629-642).

One of the benefits of such a memory-based system is a very fast computation time, both during the learning phase and during classification. For the known types of n-tuple networks, which is also known as "RAM networks" or "weightless neural networks", learning is accomplished by recording features of patterns in a random-access memory (RAM), which requires just one presentation of the training set(s) to the system.

The training procedure for a conventional RAM based neural network is described by Jørgensen (co-inventor of this invention) et al. in a contribution to a recent book on RAM based neural networks (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., World Scientific, London, pp. 78-88, 1998). The contribution describes how the RAM based neural network are considered as comprising a number of Look Up Tables (LUTs). Each LUT may probe a subset of a binary input data vector. In the conventional scheme the bits to be used are selected at random. The sampled bit sequence is used to construct an address. This address corresponds to a specific entry (column) in the LUT. The number of rows in the LUT corresponds to the number of possible classes. For each class the output will take on the values 0 or 1. A value of 1 corresponds to a vote on that specific class. When performing a classification, an input vector is sampled, the output vectors from all LUTs are added, and subsequently a winner takes all decision is made to classify the input vector. In order to perform a simple training of the network, the output values may initially be set to 0. For each example in the training set, the following steps should then be carried out:

Present the input vector and the target class to the network, for all LUTs calculate their corresponding column entries, and set the output value of the target class to 1 in all the "active" columns.

By use of such a training strategy it is guaranteed that each training pattern always obtains the maximum number of votes on the true class. As a result such a network makes no misclassification on the training set, but ambiguous decisions may occur. Here, the generalisation capability of the network is directly related to the number of input bits for each LUT. If a LUT samples all input bits then it will act as a pure memory device and no generalisation will be provided. As the number of input bits is reduced the generalisation is increased at an expense of an increasing number of ambiguous decisions. Furthermore, the classification and generalisation performances of a LUT are highly dependent on the actual subset of input bits probed. The purpose of an "intelligent" training procedure is thus to select the most appropriate subsets of input data.

Jørgensen et al. further describes what is named a "leave-one-out cross-validation test" which suggests a method for selecting an optimal number of input connections to use per LUT in order to obtain a low classification error rate with a short overall computation time. In order to perform such a cross-validation test it is necessary to obtain knowledge of the actual number of training examples that have visited or addressed the cell or element corresponding to the addressed column and class. It is therefore suggested that these numbers are stored in the LUTs. It is also suggested by Jørgensen et al. how the

LUTs in the network is selected in a more optimum way by successively training new sets of LUTs and performing cross validation test on each LUT. Thus, it is known to have a RAM network in which the LUTs are selected by presenting the training set to the system several times.

The output vector from the RAM network contains a number of output scores, one for each possible class. As mentioned above, a decision is normally made by classifying an example into the class having the largest output score. This simple winner-takes-all (WTA) scheme assures that the true class of training examples cannot lose to one of the other classes. One problem with the RAM net classification scheme is that it often behaves poorly when trained on a training set where the distribution of examples between the training classes are highly skewed. Accordingly there is a need for understanding the influence of the composition of the training material on the behaviour of the RAM classification system as well as a general understanding of the influence of specific parameters of the architecture on the performance. From such an understanding it is possible to modify the classification scheme to improve its performance and competitiveness with other schemes. Such improvements of the RAM based classification systems are provided according to the present invention.

Any discussion of the prior art throughout the specification should in no way be considered as an admission that such prior art is widely known or forms part of common general knowledge in the field.

SUMMARY OF THE INVENTION

It is an object of the present invention to overcome or ameliorate at least one of the disadvantages of the prior art, or to provide a useful alternative.

According to one aspect of the invention there is provided a method of training a computer classification system which is defined by a network comprising a number of n-tuples or Look Up Tables (LUTs), with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of possible classes and further comprising a number of columns being addressed by signals or elements of sampled training input data examples, each column is defined by a vector having cells with values, wherein

the column vector cell values are determined based on one or more training sets of input data examples for different classes so that at least part of the cells comprise or point to information based on the number of times the corresponding cell address is sampled from one or more sets of training input examples, said method being characterised in that

one or more output score functions for evaluation of at least one output score per class, and

one or more decision rules to be used in combination with at least part of the obtained output score values to determine a winning class, wherein said determination of the output score functions and decision rules comprises

determining output score functions based on the information of at least part of the determined column vector cell values, and adjusting at least part of the output score functions based on a information measure evaluation, and/or determining decision rules based on the information of at least part of the determined column vector cell values, and adjusting at least part of the decision rules based on the information measure evaluation.

Preferably, the adjustment process comprises the further steps of:

determining a global quality value based on at least part of the column vector cell values,

determining if the global quality value fulfils a required quality criterion, and

adjusting at least part of the output score functions until the global quality criterion is fulfilled.

According to a second aspect of the invention there is provided a system for training a computer classification system which is defined by the network comprising a stored number of n-tuples or LUTs, with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of possible classes and further comprising a number of columns being addressed by signals or elements of sampled training input data, each column being defined by a vector having cells with values, said system comprising:

a) input means for receiving training data examples of known classes,
b) means for sampling the received input data examples and addressing column vectors in a stored set of n-tuples or LUTs,
c) means for addressing specific rows in the set of n-tuples or LUTs, said rows corresponding to a known class,
d) storage means for storing determined n-tuples or LUTs,
e) means for determining column vector cell values so as to comprise or point to information based on the number of times the corresponding cell address

is sampled from the training set(s) of input examples, characterised in that said system further comprises

f) means for determining one or more output score functions and/or one or more decision rules, wherein said output score functions and decision rules determining means is adapted for

determining said output score functions based on the information of at least part of the determined column vector cell values and a validation set of input data examples of known classes, and

determining said decision rules based on the information of at least part of the determined column vector cell values and a validation set of input data examples of known classes, and wherein the means for determining the output score functions and decision rules comprises

means for initialising one or more sets of output score functions and/or decision rules, and

means for adjusting output score functions and decision rules by use of at least part of the validation set of input examples.

According to another aspect of the invention there is provided a system for classifying input data examples of unknown classes into at least one of a plurality of classes, said system comprising:

storage means for storing a number or set of n-tuples or LUTs with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of possible classes and further comprising a number of column vectors, each column vector being addressed by signals or elements of sampled input data example, and each column vector having cell values being determined during a training process based on one or more sets of training input data examples,

storage means for storing one or more output score functions and/or one or more decision rules, each output score function and/or decision rule being determined during a training or validation process based on one or more sets of validation input data examples, said system further comprising:

input means for receiving an input data example to be classified,

means for sampling the received input data example and addressing column vectors in the stored set of n-tuples or LUTs,

means for addressing specific rows in the set of n-tuples or LUTs, said rows corresponding to a specific class,

means for determining output score values using the stored output score functions and at least part of the stored column vector values, and

means for determining a winning class or classes based on the output score values and stored decision rules.

Recently, Thomas Martine Jorgensen and Christian Linneberg (inventors of the invention) have provided a statistical framework that have made it possible to make a theoretical analysis that relates the expected output scores of the n-tuple net to stochastic parameters of the example distributions, the number of training examples, and the number of address lines *n* used for each LUT or n-tuple. From the obtained expressions, they have been able to study the behaviour of the architecture in different scenarios. Furthermore, they have based on the theoretical results come up with proposals for modifying the n-tuple classification scheme in order to make it operate as a close approximation to the maximum a posteriori or a maximum likelihood estimator. The resulting modified decision criteria willfor example deal with the so-called skewed class prior problem causing the n-tuple net to often behave poorly when trained on a training set where the distribution of examples between the training classes are highly skewed. Accordingly the proposed changes of the classification scheme provide an essential improvement of the architecture. The suggested changes in decision criteria are not only applicable to the original n-tuple architecture based on random memorisation. It also applies to extended n-tuple schemes, some of which are more optimal selection of the address lines and some of which apply an extended weight scheme.

A preferred embodiment includes a means of adjusting output score functions and decision rules which:

a) determines a local quality value corresponding to a sampled validation input example, the local quality value being a function of at least part of the addressed vector cell values,

b) determines if the local quality value fulfils a required local quality criterion,

c) adjusts one or more of the output score functions and/or decision rules if the local quality criterion is not fulfilled,

d) repeats the local quality test for a predetermined number of training input examples,

e) determines a global quality value based on at least part of the column vectors being addressed during a local quality test,

f) determines if the global quality value fulfils a required global quality criterion, and

g) repeats the local and the global quality test until the global quality criterion is fulfilled.

Unless the context clearly requires otherwise, throughout the description and the claims, the words 'comprise', 'comprising', and the like are to be construed in an inclusive sense as opposed to an exclusive or exhaustive sense; that is to say, in the sense of "including, but not limited to".

BRIEF DESCRIPTION OF THE DRAWINGS

A preferred embodiment of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Fig. 1 shows a block diagram of a RAM classification network with Look Up Tables (LUTs),

Fig. 2 shows a detailed block diagram of a single Look Up Table (LUT) according to an embodiment of the present invention,

Fig. 3 shows a block diagram of a computer classification system according to the present invention,

Fig. 4 shows a flow chart of a learning process for LUT column cells according to an embodiment of the present invention,

THE NEXT PAGE IS PAGE 14

function of the number of times the corresponding cell address is sampled from the training set(s) of input examples.

According to an embodiment of the present invention it is preferred that when a training input data example belonging to a known class is applied to the classification network thereby addressing one or more column vectors, the means for determining the column vector cell values is adapted to increment the value or vote of the cells of the addressed column vector(s) corresponding to the row(s) of the known class, said value preferably being incremented by one.

For the adjustment process of the output score functions and decision rules it is preferred that the means for adjusting output score functions and/or decision rules is adapted to

determine a global quality value based on at least part of column vector cell values,

determine if the global quality value fulfils a required global quality criterion, and

adjust at least part of the output score functions and/or decision rules until the global quality criterion is fulfilled.

As an example of a preferred embodiment according to the present invention, the means for adjusting output score functions and decision rules may be adapted to

a) determine a local quality value corresponding to a sampled validation input example, the local quality value being a function of at least part of the addressed vector cell values,

b) determine if the local quality value fulfils a required local quality criterion,

c) adjust one or more of the output score functions and/or decision rules if the local quality criterion is not fulfilled,

d) repeat the local quality test for a predetermined number of training input examples,

e) determine a global quality value based on at least part of the column vectors being addressed during the local quality test,

f) determine if the global quality value fulfils a required global quality criterion, and,

g) repeat the local and the global quality test until the global quality criterion is fulfilled.

The means for adjusting the output score functions and decision rules may further be adapted to stop the iteration process if the global quality criterion is not fulfilled after a given number of iterations. In a preferred embodiment, the means for storing n-tuples or LUTs comprises means for storing adjusted output score functions and decision rules and separate means for storing best so far output score functions and decision rules or best so far classification system configuration values. Here, the means for adjusting the output score functions and decision rules may further be adapted to replace previously separately stored best so far output score functions and decision rules with obtained adjusted output score functions and decision rules if the determined global quality value is closer to fulfil the global quality criterion than the global quality value corresponding to previously separately stored best so far output score functions and decision rules. Thus, even if the system should not be able to fulfil the global quality criterion within a given number of iterations, the system may always comprise the "best so far" system configuration.

According to a further aspect of the present invention there is also provided a system for classifying input data examples of unknown classes into at least one of a plurality of classes, said system comprising:

storage means for storing a number or set of n-tuples or Look Up Tables (LUTs) with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of the number of possible classes and further comprising a number of column vectors, each column vector being addressed by signals or elements of a sampled input data example, and each column vector having cell values being determined during a training process based on one or more sets of training input data examples,

storage means for storing one ore more output score functions and/or one or more decision rules, each output score function and/or decision rule being determined during a training or validation process based on one or more sets of validation input data examples, said system further comprising:

input means for receiving an input data example to be classified, means for sampling the received input data example and addressing column vectors in the stored set of n-tuples or LUTs,

means for addressing specific rows in the set of n-tuples or LUTs, said rows corresponding to a specific class,

means for determining output score values using the stored output score functions and at least part of the stored column vector values, and

means for determining a winning class or classes based on the output score values and stored decision rules.

It should be understood that it is preferred that the cell values of the column vectors and the output score functions and/or decision rules of the classification system according to the present invention are determined by use of a training system according to any of the above described systems. Accordingly, the column vector cell values and the output score functions and/or decision rules may be determined during a training process according to any of the above described methods.

BRIEF DESCRIPTION OF THE DRAWINGS

For a better understanding of the present invention and in order to show how the same may be carried into effect, reference will now be made by way of example to the accompanying drawings in which:

Fig. 1 shows a block diagram of a RAM classification network with Look Up Tables (LUTs),

Fig. 2 shows a detailed block diagram of a single Look Up Table (LUT) according to an embodiment of the present invention,

Fig. 3 shows a block diagram of a computer classification system according to the present invention,

Fig. 4 shows a flow chart of a learning process for LUT column cells according to an embodiment of the present invention,

Fig. 5 shows a flow chart of a learning process according to a embodiment of the present invention,

Fig. 6 shows a flow chart of a classification process according to the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following a more detailed description of the architecture and concept of a classification system according to the present invention will be given including an example of a training process of the column cells of the architecture and an example of a classification process. Furthermore, different examples of learning processes for the output score functions and the decision rules according to embodiments of the present invention are described.

Notation

The notation used in the following description and examples is as follows:

X:	The training set.
$\bar{x}$:	An example from the training set.
N_X:	Number of examples in the training set X.
$\bar{x}_j$:	The j'th example from a given ordering of the training set X.
$\bar{y}$:	A specific example (possible outside the training set).
C:	Class label.
$C(\bar{x})$:	Class label corresponding to example $\bar{x}$ (the true class).
C_W:	Winner Class obtained by classification.
C_T:	True class obtained by classification.
N_C:	Number of training classes corresponding to the maximum number of rows in a LUT.
Ω:	Set of LUTs (each LUT may contain only a subset of all possible address columns, and the different columns may register only subsets of the existing classes).
N_{LUT}:	Number of LUTs.

N_{COL}:	Number of different columns that can be addressed in a specific LUT (LUT dependent).
X_C:	The set of training examples labelled class C.
v_{iC}:	Entry counter for the cell addressed by the i'th column and the C'th class.
$a_i(\bar{y})$:	Index of the column in the i'th LUT being addressed by example $\bar{y}$.
$\bar{v}$:	Vector containing all v_{iC} elements of the LUT network.
Q_L:	Local quality function.
Q_G:	Global quality function.
$\mathbf{B}^{C_i,C_j}$:	Decision rule matrix
M_{c_i,c_j}:	Cost matrix
$S\cdot$:	Score function
$\Gamma\cdot$:	Leave-one-out cross-validation score function
$\mathbf{P}$:	Path matrix
$\bar{\beta}$:	Parameter vector
Ξ:	Set of decision rules
d_c:	Score value on class c
$D(\cdot)$:	Decision function

Description of architecture and concept

In the following references are made to Fig. 1, which shows a block diagram of a RAM classification network with Look Up Tables (LUTs), and Fig. 2, which shows a detailed block diagram of a single Look Up Table (LUT) according to an embodiment of the present invention.

A RAM-net or LUT-net consists of a number of Look Up Tables (LUTs) (1.3). Let the number of LUTs be denoted N_{LUT}. An example of an input data vector $\bar{y}$ to be classified may be presented to an input module (1.1) of the LUT network. Each LUT may sample a part of the input data, where different numbers of input signals may be sampled for different LUTs (1.2) (in principle it is also possible to have one LUT sampling the whole input space). The outputs of the LUTs may be fed (1.4) to an output module (1.5) of the RAM classification network.

In Fig. 2 it is shown that for each LUT the sampled input data (2.1) of the example presented to the LUT-net may be fed into an address selecting module (2.2). The address selecting module (2.2) may from the input data calculate the address of one or more specific columns (2.3) in the LUT. As an example, let the index of the column in the i'th LUT being addressed by an input example $\bar{y}$ be calculated as $a_i(\bar{y})$. The number of addressable columns in a specific LUT may be denoted N_{COL}, and varies in general from one LUT to another. The information stored in a specific row of a LUT may correspond to a specific class C (2.4). The maximum number of rows may then correspond to the number of classes, N_C. The number of cells within a column corresponds to the number of rows within the LUT. The column vector cells may correspond to class specific entry counters of the column in question. The entry counter value for the cell addressed by the i'th column and class C is denoted v_{iC} (2.5).

The v_{iC}-values of the activated LUT columns (2.6) may be fed (1.4) to the output module (1.5), where one or more output scores may be calculated for each class and where these output scores in combinations with a number of decision rules determine the winning class.

Let $\bar{x} \in X$ denote an input data example used for training and let $\bar{y}$ denote an input data example not belonging to the training set. Let $C(\bar{x})$ denote the class to which $\bar{x}$ belongs. The class assignment given to the example $\bar{y}$ is then obtained by calculating one or more output scores for each class. The output scores obtained for class C is calculated as functions of the v_{iC} numbers addressed by the example $\bar{y}$ but will in general also depend on a number of parameters $\bar{\beta}$. Let the mth output score of class C be denoted $S_{C,m}(v_{iC},\bar{\beta})$. A classification is obtained by combining the obtained output scores from all classes with a number of decision rules. The effect of the decision rules is to define regions in the output score space that must be addressed by the output score values to obtain a given winner class. The set of decision rules is denoted Ξ and corresponds to a set of decision borders.

Figure 3 shows an example of a block diagram of a computer classification system according to the present invention. Here a source such as a video camera or a database provides an input data signal or signals (3.0) describing the example to be classified. These data are fed to a pre-processing module (3.1) of a type which can extract features, reduce, and transform the input data in a predetermined manner. An example of such a pre-processing module is a FFT-board (Fast Fourier Transform). The transformed data are then fed to a classification unit (3.2) comprising a RAM network according to the present invention. The classification unit (3.2) outputs a ranked classification list which might have associated confidences. The classification unit can be implemented by using software to programme a standard Personal Computer or programming a hardware device, e.g. using programmable gate arrays combined with RAM circuits and a digital signal processor. These data can be interpreted in a post-processing device (3.3), which could be a computer module combining the obtained classifications with other relevant information. Finally the result of this interpretation is fed to an output device (3.4) such as an actuator.

Initial training of the architecture

The flow chart of Fig. 4 illustrates a one pass learning scheme or process for the determination of the column vector entry counter or cell distribution, v_{iC}-distribution (4.0), according to an embodiment of the present invention, which may be described as follows:

1. Initialise all entry counters or column vector cells by setting the cell values, $\bar{v}$, to zero (4.1).
2. Present the first training input example, $\bar{x}_1$ from the training set X to the network (4.2, 4.3).
3. Calculate the columns addressed for the first LUT (4.4, 4.5).
4. Add 1 to the entry counters in the rows of the addressed columns that correspond to the class label of $\bar{x}$ (increment $v_{a_i(\bar{x}),C(\bar{x})}$ in all LUTs) (4.6).
5. Repeat step 4 for the remaining LUTs (4.7, 4.8).
6. Repeat steps 3-5 for the remaining training input examples (4.9, 4.10). The number of training examples is denoted N_X.

Initialisation of output score functions and decision rules

Before the trained network can be used for classification the output score functions and the decision rules must be initialised.

Classification of an unknown input example

When the RAM network of the present invention has been trained to thereby determine values for the column cells whereby the LUTs may be defined, the network may be used for classifying an unknown input data example.

In a preferred example according to the present invention, the classification is performed by using the decision rules Ξ and the output scores obtained from the output score functions. Let the decision function invoking Ξ and the output scores be denoted $D(\cdot)$. The winning class can then be written as:

$$\text{Winner Class} = D(\Xi,\ S_{1,1}, S_{1,2}, \ldots S_{1,j} \ldots S_{2,1} \ldots\ldots S_{2,k}, \ldots S_{l,m})$$

Figure 6 shows a block diagram of the operation of a computer classification system in which a classification process (6.0) is performed. The system acquires one or more input signals (6.1) using e.g. an optical sensor system. The obtained input data are preprocessed (6.2) in a pre-processing module, e.g. a low-pass filter, and presented to a classification module (6.3) which according to an embodiment of the invention may be a LUT-network. The output data from the classification module is then post-processed in a post-processing module (6.4), e.g. a CRC algorithm calculating a cyclic redundancy check sum, and the result is forwarded to an output device (6.5), which could be a monitor screen.

Adjustment of output score function parameter $\bar{\beta}$ and adjustment of decision rules Ξ

Usually the initially determined values of $\bar{\beta}$ and the initial set of rules Ξ will not present the optimal choices. Thus, according to a preferred embodiment of the present invention, an optimisation or adjustment of the $\bar{\beta}$ values and the Ξ rules should be performed.

In order to select or adjust the parameters $\bar{\beta}$ and the rules Ξ to improve the performance of the classification system, it is suggested according to an embodiment of the invention to define proper quality functions for measuring the performance of the $\bar{\beta}$-values and the Ξ- rules. Thus, a local quality function $Q_L(\bar{v},\bar{x},X,\bar{\beta},\Xi)$ may be defined, where $\bar{v}$ denotes a vector containing all v_{iC} elements of the LUT network. The local quality function may give a confidence measure of the output classification of a specific example $\bar{x}$. If the quality value does not satisfy a given criterion the $\bar{\beta}$ values and the Ξ rules are adjusted to make the quality value satisfy or closer to satisfying the criterion (if possible).

Furthermore a global quality function: $Q_G(\bar{v},X,\bar{\beta},\Xi)$ may be defined. The global quality function may measure the performance of the input training set as a whole.

Fig. 5 shows a flow chart for adjustment or learning of the $\bar{\beta}$ values and the Ξ rules according to the present invention.

Example 1

This example illustrates an optimisation procedure for adjusting the decision rules Ξ.

We consider N_c training classes. The class label c is an integer running from 1 to N_c.

For each class c we define a single output score function:

$$S_c\left(v_{a,(\bar{x}),c},\bar{\beta}\right) = \sum_{l\in\Omega} \beta_l \Theta_k\left(v_{a,(\bar{x}),c}\right),\ \bar{\beta} = (\beta_1,\beta_2,\ldots)$$

where $\delta_{i,j}$ is Kroneckers delta ($\delta_{i,j} = 1$ if $i = j$ and 0 otherwise), and

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

The expression for the output score function illustrates a possible family of functions determined by a parameter vector $\bar{\beta}$. This example, however, will only illustrate a procedure for adjusting the decision rules Ξ, and not $\bar{\beta}$. For simplicity of notation we therefore initialise all values in $\bar{\beta}$ to one. We then have:

$$S_c(v_{a,(x),c}) = \sum_{i\in\Omega} \Theta_k(v_{a,(x),c}).$$

With this choice of $\bar{\beta}$ the possible output values for S_c are the integers from 0 to N_{LUT} (both inclusive).

The leave-one-out cross-validation score or vote-count on a given class c is:

$$\Gamma_c(\bar{x}) = \sum_{i\in\Omega} \Theta_{k+\delta_{C_T(\bar{x}),c}}(v_{a,(\bar{x}),c}),$$

where $C_T(\bar{x})$ denotes the true class of example $\bar{x}$.

For all possible inter-class combinations (c_1,c_2), $(c_1 \in \{1,2,\dots N_c\}, c_2 \in \{1,2,\dots N_c\}) \wedge (c_1 \neq c_2)$ we wish to determine a suitable decision border in the score space spanned by the two classes. The matrix $\mathbf{B}^{c_1,c_2}$ is defined to contain the decisions corresponding to a given set of decision rules applied to the two corresponding output score values; i.e whether class c_1 or class c_2 wins. The row and column dimensions are given by the allowed ranges of the two output score values, i.e. the matrix dimension is $(N_{LUT}+1)\times(N_{LUT}+1)$. Accordingly, the row and column indexes run from 0 to N_{LUT}.

Each matrix element contains one of the following three values: c_1, c_2 and k_{AMB}, where k_{AMB} is a constant different from c_1 and c_2. Here we use $k_{AMB} = 0$. The two output score values S_1 and S_2 obtained for class c_1 and class c_2, respectively, are used to address the element $b^{c_1,c_2}_{S_1,S_2}$ in the matrix $\mathbf{B}^{c_1,c_2}$. If the addressed element contains the value c_1 it means that class c_1 wins over class c_2. If the addressed element contains the value c_2 it means that class c_2 wins over class c_1. Finally, if the addressed element contains the value k_{AMB}, it means the decision is ambiguous.

The decision rules are initialised to correspond to a WTA decision. This corresponds to having a decision border along the diagonal in the matrix $\mathbf{B}^{c_1,c_2}$. Along the diagonal the elements are initialised to take on the value k_{AMB}. Above and respectively below the diagonal the elements are labelled with opposite class values.

A strategy for adjusting the initialised decision border according to an information measure that uses the $v_{a_i(\bar{x}),c}$ values is outlined below.

Create the cost matrix $\mathbf{M}^{c_1,c_2}$ with elements given as:

$$m_{i,j} = \alpha_{c_1,c_2} \sum_{\bar{x} \in \mathbf{X}_{c_1}} \left(\Gamma_{c_1}(\bar{x}) \le i \wedge \Gamma_{c_2}(\bar{x}) \ge j\right) + \alpha_{c_2,c_1} \sum_{\bar{x} \in \mathbf{X}_{c_2}} \left(\Gamma_{c_1}(\bar{x}) \ge i \wedge \Gamma_{c_2}(\bar{x}) \le j\right)$$

α_{c_1,c_2} denotes the cost associated with classifying an example from class c_1 in to class c_2 and α_{c_2,c_1} denotes the cost associated with the opposite error. It is here assumed that a logical true evaluates to one and a logical false evaluates to zero.

A minimal-cost path from $m_{0,0}$ to $m_{N_{LUT},N_{LUT}}$ can be calculated using e.g. a dynamic programming approach as shown by the following pseudo-code: (the code uses a path matrix $\mathbf{P}^{c_1,c_2}$ with the same dimensions as $\mathbf{B}^{c_1,c_2}$)

```
// Loop through all entries in the cost matrix in reverse order:

for i := NLUT to 0 step -1
{
  for j := NLUT to 0 step -1
  {
    if ((i <> NLUT) and (j <> NLUT))
    {
      // For each entry, calculate the lowest
```

```
        // associated total-costs given as
        m_{i,j} := m_{i,j} + min(m_{i+1,j}, m_{i+1,j+1}, m_{i,j+1});
        // (Indexes outside the matrix are considered
        // as addressing the value of infinity)

        if ( min(m_{i+1,j}, m_{i+1,j+1}, m_{i,j+1}) = = m_{i+1,j} ) p_{i,j} := 1;
        if ( min(m_{i+1,j}, m_{i+1,j+1}, m_{i,j+1}) = = m_{i+1,j+1} ) p_{i,j} := 2;
        if ( min(m_{i+1,j}, m_{i+1,j+1}, m_{i,j+1}) = = m_{i,j+1} ) p_{i,j} := 3;
      }
    }
}

//According to the dynamic programming approach the path
//with the smallest associated total-cost is now obtained
//by traversing the P-matrix in the following manner to obtain
//the decision border in the score space spanned by the
//classes in question.

i := 0;
j := 0;

repeat
{
    b_{i,j}^{c1,c2} := 0;
    for a := i+1 to N_LUT step 1
    {
        b_{a,j}^{c1,c2} := c1;
    }
    for a := j+1 to N_LUT step 1
    {
        b_{i,a}^{c1,c2} := c2;
    }
```

```
        iold := i;
        jold := j;

        if (p_iold,jold < 3) then i := iold + 1;
        if (p_iold,jold > 1) then j := jold + 1;
    } until (i == N_LUT and j == N_LUT);
```

The dynamic programming approach can be extended with regularisation terms, which constraint the shape of the border.
An alternative method for determining the decision border could be to fit a *B-spline* with two control points in such a way that the associated cost is minimised.

Using the decision borders determined from the strategy outlined above an example can now be classified in the following manner:

- Present the example to the network in order to obtain the score values or vote numbers $S_c(\bar{x}) = \sum_{i \in \Omega} \Theta_k(v_{a_i(x),c})$
- Define a new set of score values d_c for all classes and initialise the scores to zero: $d_c = 0,\ 1 \le c \le N_c$.
- Loop through all possible inter-class combinations, (c_1, c_2), and update the vote-values: $d_{b^{c_1 c_2}_{S_{c_1}(\bar{x}),S_{c_2}(\bar{x})}} := d_{b^{c_1 c_2}_{S_{c_1}(\bar{x}),S_{c_2}(\bar{x})}} + 1$
- The example is now classified as belonging to the class with the label found from $\underset{c}{\operatorname{argmax}}(d_c)$.

A leave-one-out cross-validation test using the decision borders determined from the strategy outlined above is obtained in the following manner:

- Present the example to the network in order to obtain the leave-one-out score values or vote numbers $\Gamma_c(\bar{x}) = \sum_{i \in \Omega} \Theta_{k+\delta_{c_{T(i)},c}}(v_{a_i(x),c})$
- Define a new set of score values d_c for all classes and initialise the scores to zero: $d_c = 0,\ 1 \le c \le N_c$.

- Loop through all possible inter-class combinations, (c_1, c_2), and update the vote-values: $d_{b^{c_1,c_2}_{r_{c_1}(\bar{x}), r_{c_2}(\bar{x})}} := d_{b^{c_1,c_2}_{r_{c_1}(\bar{x}), r_{c_2}(\bar{x})}} + 1$
- The example is now classified as belonging to the class with the label found from $\underset{c}{\operatorname{argmax}}(d_c)$.

With reference to Figure 5 the above adjustment procedure for the decision rules (borders) Ξ may be described as

- Initialise the system by setting all values of $\bar{\beta}$ to one, selecting a WTA scheme on a two by two basis and by training the n-tuple classifier according to the flow chart in Fig. 4. (5.0)
- Batch mode optimisation is chosen. (5.1)
- Test all examples by performing a leave-one-out classification as outline above (5.12) and calculate the obtained leave-one-out cross-validation error rate and use it as the Q_G-measure. (5.13)
- Store the values of $\bar{\beta}$ and the corresponding Q_G-value as well as the Ξ-rules (the $\mathbf{B}^{c_1,c_2}$ matrices). (5.14)
- If the Q_G-value does not satisfy a given criterion or another stop criterion is met then adjust the Ξ-rules according to the dynamic programming approach outline above. (5.16, 5.15)
- If the Q_G-value is satisfied or another stop criterion is met then select the combination with the lowest total error-rate. (5.17)

In the above case one would as alternative stop criterion use a criterion that only allows two loops through the adjustment scheme.

Example 2

This example illustrates an optimisation procedure for adjusting $\bar{\beta}$.

For each class we again define a single output score

$$S_c\left(v_{a,(x),c}, \bar{\beta}\right) = \sum_{i \in \Omega} \Theta_{k_i}\left(v_{a,(x),c}\right).$$

With these score values the example is now classified as belonging to the class with the label found from $\underset{c}{\operatorname{argmax}}(S_c)$.

In this example we use $\bar{\beta} = (k_1, k_2, \ldots, k_{N_c})$. We also initialise the Ξ rules to describe a WTA decision when comparing the output scores from the different classes.

- Initialise the system by setting all k_c-values to one, selecting a WTA scheme and by training the n-tuple classifier according to the flow chart in Fig. 4. (5.0)
- Batch mode optimisation is chosen. (5.1)
- Test all examples using a leave-one-out cross-validation test (5.12) and calculate the obtained leave-one-out cross-validation error rate used as Q_G. (5.13)
- Store the values of $\bar{\beta}$ and the corresponding Q_G value. (5.14)
- Loop through all possible combinations of $k_{c_1}, k_{c_2}, \mathrm{K}, k_{c_{N_c}}$ where $k_j \in \{1,2,3,\ldots k_{MAX}$. (5.16, 5.15)
- Select the combination with the lowest total error-rate. (5.17)

For practical use, the k_{MAX}-value will depend upon the skewness of the class priors and the number of address-lines used in the RAM net system.

Example 3

This example also illustrates an optimisation procedure for adjusting $\bar{\beta}$ but with the use of a local quality function Q_L.

For each class we now define as many output scores as there are competing classes, i.e. $N_c - 1$ output scores:

$$S_{c_j, c_k}\left(v_{a_i(\bar{x}), c_j}, \bar{\beta}\right) = \sum_{i \in \Omega} \Theta_{k_{c_j, c_k}}\left(v_{a_i(\bar{x}), c_j}\right), \ \forall k \neq j.$$

With these score values a decision is made in the following manner

- Define a new set of score values d_c for all classes and initialise the scores to zero: $d_c = 0,\ 1 \le c \le N_c$.
- Loop through all possible inter-class combinations, (c_1, c_2), and update the vote-values:

  If $S_{c_1,c_2} > S_{c_2,c_1}$ then $d_{c_1} := d_{c_1} + 1$ else $d_{c_2} := d_{c_2} + 1$.
- The example is now classified as belonging to the class with the label found from $\underset{c}{\operatorname{argmax}}(d_c)$.

In this example we use

$$\bar{\beta} = (k_{c_1,c_2}, k_{c_1,c_3}, \ldots, k_{c_1,c_{N_c-1}}, k_{c_2,c_1}, \ldots, k_{c_{N_c},c_{N_c-1}}).$$

We also initialise the Ξ rules to describe a WTA decision when comparing the output scores from the different classes.

- Initialise the system by setting all k_{c_1,c_2}-values to say two, selecting a WTA scheme and by training the n-tuple classifier according to the flow chart in Fig. 4. (5.0)
- On line mode as opposed to batch mode optimisation is chosen. (5.1)
- For all examples in the training set (5.2, 5.7, and 5.8) do:

- Test each example to obtain the winner class C_W in a leave-one-crossvalidation. Let the Q_L- measure compare C_W with the true class C_T. (5.3,5.4)
- If $C_W \neq C_T$ a leave-one-out error is made so the values of k_{C_W,C_T} and k_{C_T,C_W} are adjusted by incrementing k_{C_W,C_T} with a small value, say 0.1, and by decrementing k_{C_T,C_W} with a small value, say 0.05. If the adjustment will bring the values below one, no adjustment is performed. (5.5,5.6)

- When all examples have been processed the global information measure Q_G (e.g. the leave-one-out-error-rate) is calculated and the values of $\bar{\beta}$ and Q_G are stored. (5.9,5,10)

- If Q_G or another stop criterion is not fulfilled the above loop is repeated. (5.11)
- If Q_G is satisfied or another stop criterion is fulfilled the best value of the stored Q_G-values are chosen together with the corresponding parameter values $\bar{\beta}$ and decision rules Ξ. (5.17,5.18)

The foregoing description of preferred exemplary embodiments of the invention has been presented for the purpose of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed, and obviously many modifications and variations are possible in light of the present invention to those skilled in the art. All such modifications which retain the basic underlying principles disclosed and claimed herein are within the scope of this invention.

THE CLAIMS DEFINING THE INVENTION ARE AS FOLLOWS:-

1. A method of training a computer classification system which is defined by a network comprising a number of n-tuples or Look Up Tables (LUTs), with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of possible classes and further comprising a number of columns being addressed by signals or elements of sampled training input data examples, each column being defined by a vector having cells with values, wherein

the column vector cell values are determined based on one or more training sets of input data examples for different classes so that at least part of the cells comprise or point to information based on the number of times the corresponding cell address is sampled from one or more sets of training input examples, said method being characterised in that

one or more output score functions are determined for evaluation of at least one output score value per class, and

one or more decision rules are determined to be used in combination with at least part of the obtained output scores to determine a winning class, wherein said determination of the output score functions and decision rules comprises

determining output score functions based on the information of at least part of the determined column vector cell values, and adjusting at least part of the output score functions based on an information measure evaluation, and/or

determining decision rules based on the information of at least part of the determined column vector cell values, and adjusting at least part of the decision rules based on an information measure evaluation.

2. A method according to claim 1, wherein the output score functions are determined based on a validation set of input data examples.

3. A method according to claim 1 or claim 2, wherein the decision rules are determined based on a validation set of input data examples.

4. A method according to any one of claims 1 to 3, wherein determination of the output score functions is based on an information measure evaluating the performance on the validation example set.

5. A method according to any one of claims 1 to 4, wherein determination of the decision rules is based on an information measure evaluating the performance on the validation example set.

6. A method according to any one of claims 3 to 5, wherein the validation example set equals at least part of the training set and the information measure is based on a leave-one-out cross validation evaluation.

7. A method according to any one of claims 3 to 6, wherein the validation set comprises at least part of the training set(s) of input data examples.

8. A method according to any one of claims 1 to 7, wherein the output score functions are determined by a set of parameter values.

9. A method according to any one of claims 1 to 8, wherein determination of the output score functions comprises initialising the output score functions.

10. A method according to claim 9, wherein the initialisation of the output score functions comprises determining a number of set-up parameters.

11. A method according to claims 9 or 10, wherein the initialisation of the output score functions comprises setting all output score functions to a pre-determined mapping function.

12. A method according to any one of claims 1 to 11, wherein determination of the decision rules comprises initialising the decision rules.

13. A method according to claim 12, wherein the initialisation of the decision rules comprises setting the rules to a pre-determined decision scheme.

14. A method according to any one of claims 10 to 13, wherein the adjustment comprises changing the values of the set-up parameters.

15. A method according to any one of claims 1 to 14, wherein the determination of the column vector cell values comprises the training steps of

a) applying a training input data example of a known class to the classification network, thereby addressing one or more column vectors,

b) incrementing, preferably by one, the value or vote of the cells of the addressed column vector(s) corresponding to the row(s) of the known class, and

c) repeating steps (a) to (b) until all training examples have been applied to the network.

16. A method according to any one of claims 1 to 15, wherein the adjustment process comprises the steps of

determining a global quality value based on at least part of the column vector cell values,

determining if the global quality value fulfils a required quality criterion, and adjusting at least part of the output score functions until the global quality criterion is fulfilled.

17. A method according to any one of claims 1 to 16, wherein the adjustment process comprises the steps of

a) selecting an input example from the validation set(s),

b) determining a local quality value corresponding to the sampled validation input example, the local quality value being a function of at least part of the addressed column cell values,

c) determining if the local quality value fulfils a required local quality criterion, if not, adjusting one or more of the output score functions if the local quality criterion is not fulfilled,

d) selecting a new input example from a predetermined number of examples of the validation set(s),

e) repeating the local quality test steps (b) to (d) for all the predetermined validation input examples,

f) determining a global quality value based on at least part of the column vectors being addressed during the local quality test,

g) determining if the global quality value fulfils a required global quality criterion, and,

h) repeating steps (a) to (g) until the global quality criterion is fulfilled.

18. A method according to any one of claims 1 to 17, wherein the adjustment process comprises the steps of

determining a global quality value based on at least part of the column vector cell values,

determining if the global quality value fulfils a required quality criterion, and adjusting at least part of the decision rules until the global quality criterion is fulfilled.

19. A method according to any one of claims 1 to 18, wherein the adjustment process comprises the steps of

a) selecting an input example from the validation set(s),

b) determining a local quality value corresponding to the sampled validation input example, the local quality value being a function of at least part of the addressed column cell values,

c) determining if the local quality value fulfils a required local quality criterion, if not, adjusting one or more of the decision rules if the local quality criterion is not fulfilled,

d) selecting a new input example from a predetermined number of examples of the validation set(s),

e) repeating the local quality test steps (b) to (d) for all the predetermined validation input examples,

f) determining a global quality value based on at least part of the column vectors being addressed during the local quality test,

g) determining if the global quality value fulfils a required global quality criterion, and,

h) repeating steps (a) to (g) until the global quality criterion is fulfilled.

20. A method according to any one of claims 1 to 15, wherein the adjustment process comprises the steps of:

determining a global quality value based on at least part of the column vector cell values,

determining if the global quality value fulfils a required quality criterion, and

adjusting at least part of the output score functions and part of the decision rules until the global quality criterion is fulfilled.

21. A method according to any one of claims 1 to 15 or 20, wherein the adjustment process comprises the steps of

a) selecting an input example from the validation set(s),

b) determining a local quality value corresponding to the sampled validation input example, the local quality value being a function of at least part of the addressed column cell values,

c) determining if the local quality value fulfils a required local quality criterion, if not, adjusting one or more of the output score functions and the decision rules if the local quality criterion is not fulfilled,

d) selecting a new input example from a predetermined number of examples of the validation set(s),

e) repeating the local quality test steps (b) to (d) for all the predetermined validation input examples,

f) determining a global quality value based on at least part of the column vectors being addressed during the local quality test,

g) determining if the global quality value fulfils a required global quality criterion, and,

h) repeating steps (a) to (g) until the global quality criterion is fulfilled.

22. A method according to claim 17, 19 or 21, wherein steps (b) to (d) are carried out for all examples of the validation set(s).

23. A method according to any one of claims 16 to 22, wherein the local and/or global quality value is defined as functions of at least part of the column cells.

24. A method according to any one of claims 16 to 23, wherein the adjustment iteration process is stopped if the quality criterion is not fulfilled after a given number of iterations.

25. A method of classifying input data examples into at least one of a plurality of classes using a computer classification system configured according to any one of claims 1 to 24, whereby column cell values for each n-tuple or LUT and output score functions

and/or decision rules are determined using on one or more training or validation sets of input data examples, said method comprising:

a) applying an input data example to be classified to the configured classification network thereby addressing column vectors in the set of n-tuples or LUTs,

b) selecting a set of classes which are to be compared using a given set of output score functions and decision rules thereby addressing specific rows in the set of n-tuples or LUTs,

c) determining output score values as a function of the column vector cells and using the determined output score functions,

d) comparing the calculated output values using the determined decision rules, and

e) selecting the class or classes that win(s) according to the decision rules.

26. A system for training a computer classification system which is defined by a network comprising a stored number of n-tuples or Look Up Tables (LUTs), with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of possible classes and further comprising a number of columns being addressed by signals or elements of sampled training input data examples, each column being defined by a vector having cells with values, said system comprising:

a) input means for receiving training input data examples of known classes,

b) means for sampling the received input data examples and addressing column vectors in the stored set of n-tuples or LUTs,

c) means for addressing specific rows in the set of n-tuples or LUTs, said rows corresponding to a known class,

d) storage means for storing determined n-tuples or LUTs,

e) means for determining column vector cell values so as to comprise or point to information based on the number of times the corresponding cell address is sampled from the training set(s) of input examples, characterised in that said system further comprises

f) means for determining one or more output score functions and one or more decision rules, wherein said output score functions and decision rules determining means is adapted for

determining said output score functions based on the information of at least part of the determined column vector cell values and a validation set of input data examples of known classes, and

determining said decision rules based on the information of at least part of the determined column vector cell values and a validation set of input data examples of known classes, and wherein the means for determining the output score functions and decision rules comprises

means for initialising one or more sets of output score functions and/or decision rules, and

means for adjusting output score functions and decision rules by use of at least part of the validation set of input examples.

27. A system according to claim 26, wherein the means for determining the output score functions is adapted to determine such functions from a family of output score functions determined by a set of parameter values.

28. A system according to claim 26 or 27, wherein said validation set comprises at least part of the training set(s) used for determining the column cell values.

29. A system according to any one of claims 26 to 28, wherein the means for determining the column vector cell values is adapted to determine these values as a function of the number of times the corresponding cell address is sampled from the set(s) of training input examples.

30. A system according to any one of claims 26 to 29, wherein, when a training input data example belonging to a known class is applied to the classification network thereby addressing one or more column vectors, the means for determining the column vector cell values is adapted to increment the value or vote of the cells of the addressed column vector(s) corresponding to the row(s) of the known class, said value preferably being incremented by one.

31. A system according to any one of claims 26 to 30, wherein the means for adjusting output score functions is adapted to

determine a global quality value based on at least part of column vector cell values,

determine if the global quality value fulfils a required global quality criterion, and

adjust at least part of the output score functions until the global quality criterion is fulfilled.

32. A system according to any one of claims 26 to 31, wherein the means for adjusting output score functions and decision rules is adapted to:

a) determine a local quality value corresponding to a sampled validation input example, the local quality value being a function of at least part of the addressed vector cell values,

b) determine if the local quality value fulfils a required local quality criterion,

c) adjust one or more of the output score functions if the local quality criterion is not fulfilled,

d) repeat the local quality test for a predetermined number of training input examples,

e) determine a global quality value based on at least part of the column vectors being addressed during the local quality test,

f) determine if the global quality value fulfils a required global quality criterion, and,

g) repeat the local and the global quality test until the global quality criterion is fulfilled.

33. A system according to any one of claims 26 to 32, wherein the means for adjusting decision rules is adapted to

determine a global quality value based on at least part of column vector cell values, determine if the global quality value fulfils a required global quality criterion, and adjust at least part of the decision rules until the global quality criterion is fulfilled.

34. A system according to any one of claims 26 to 33, wherein the means for adjusting output score functions and decision rules is adapted to

a) determine a local quality value corresponding to a sampled validation input example, the local quality value being a function of at least part of the addressed vector cell values,

b) determine if the local quality value fulfils a required local quality criterion,

c) adjust one or more of the decision rules if the local quality criterion is not fulfilled,

d) repeat the local quality test for a predetermined number of training input examples,

e) determine a global quality value based on at least part of the column vectors being addressed during the local quality test,

f) determine if the global quality value fulfils a required global quality criterion, and,

g) repeat the local and the global quality test until the global quality criterion is fulfilled.

35. A system according to any one of claims 26 to 30, wherein the means for adjusting decision rules is adapted to

determine a global quality value based on at least part of column vector cell values, determine if the global quality value fulfils a required global quality criterion, and adjust least part of the output score functions and decision rules until the global quality criterion is fulfilled.

36. A system according to any one of claims 26 to 30 or 35, wherein the means for adjusting output score functions and decision rules is adapted to

a) determine a local quality value corresponding to a sampled validation input example, the local quality value being a function of at least part of the addressed vector cell values,

b) determine if the local quality value fulfils a required local quality criterion,

c) adjust one or more of the output score functions and decision rules if the local quality criterion is not fulfilled,

d) repeat the local quality test for a predetermined number of training input examples,

e) determine a global quality value based on at least part of the column vectors being addressed during the local quality test,

f) determine if the global quality value fulfils a required global quality criterion, and,

g) repeat the local and the global quality test until the global quality criterion is fulfilled.

37. A system according to any one of claims 31 to 36, wherein the means for adjusting the output score functions and decision rules is further adapted to stop the iteration process if the global quality criterion is not fulfilled after a given number of iterations.

38. A system according to any one of claims 26 to 37, wherein the means for storing n-tuples or LUTs comprises means for storing adjusted output score functions and decision rules and separate means for storing best so far output score functions and decision rules or best so far classification system configuration values.

39. A system according to claim 38, wherein the means for adjusting the output score functions and decision rules is further adapted to replace previously separately stored

best so far output score functions and decision rules with obtained adjusted output score functions and decision rules if the determined global quality value is closer to fulfil the global quality criterion than the global quality value corresponding to previously separately stored best so far output score functions and decision rules.

40. A system for classifying input data examples of unknown classes into at least one of a plurality of classes, said system comprising:

storage means for storing a number or set of n-tuples or Look Up Tables (LUTs) with each n-tuple or LUT comprising a number of rows corresponding to at least a subset of the number of possible classes and further comprising a number of column vectors, each column vector being addressed by signals or elements of a sampled input data example, and each column vector having cell values being determined during a training process based on one or more sets of training input data examples,

storage means for storing one ore more output score functions and/or one or more decision rules, each output score function and/or decision rule being determined during a training or validation process based on one or more sets of validation input data examples, said system further comprising:

input means for receiving an input data example to be classified,

means for sampling the received input data example and addressing column vectors in the stored set of n-tuples or LUTs,

means for addressing specific rows in the set of n-tuples or LUTs, said rows corresponding to a specific class,

means for determining output score values using the stored output score functions and at least part of the stored column vector values, and means for determining a winning class or classes based on the output score values and stored decision rules.

41. A system according to claim 40, wherein the cell values of the column vectors and the output score functions and/or decision rules of the classification system are determined by use of a training system according to any one of claims 26 to 39.

42. A system according to claim 40, wherein the column vector cell values and the output score functions and/or decision rules are determined during a training process according to any one of claims 1 to 24.

43. A method of training a computer classification system substantially as herein described with reference to any one of the embodiments of the invention illustrated in the accompanying drawings and/or examples.

44. A system for training a computer classification system substantially as herein described with reference to any one of the embodiments of the invention illustrated in the accompanying drawings and/or examples.

45. A system for classifying input data samples substantially as herein described with reference to any one of the embodiments of the invention illustrated in the accompanying drawings and/or examples.

DATED this 30th Day of July, 2001

INTELLIX A/S

Attorney: JOHN B. REDFERN
Fellow Institute of Patent and Trade Mark Attorneys of Australia
of BALDWIN SHELSTON WATERS

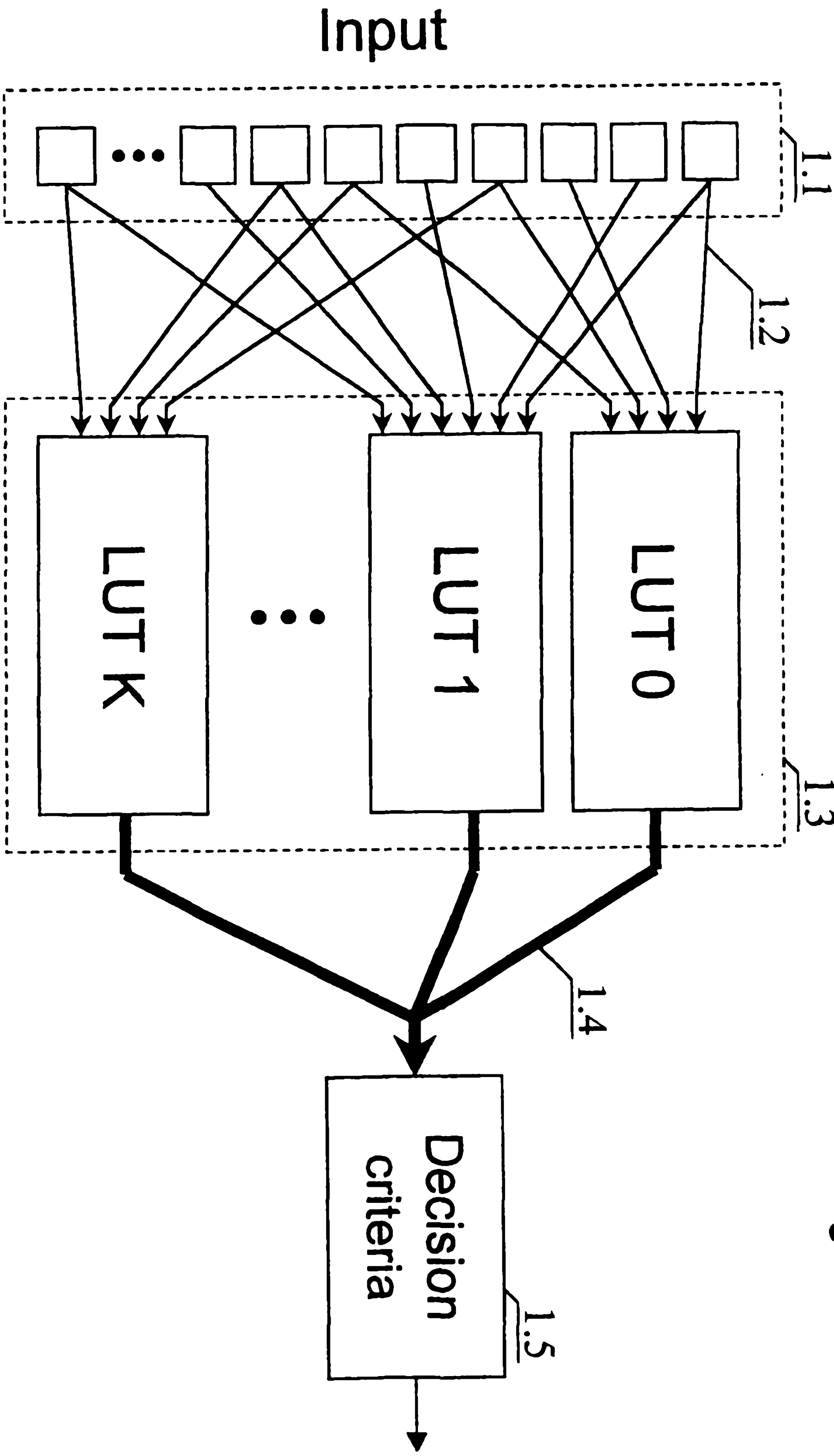


Figure 1

Figure 2

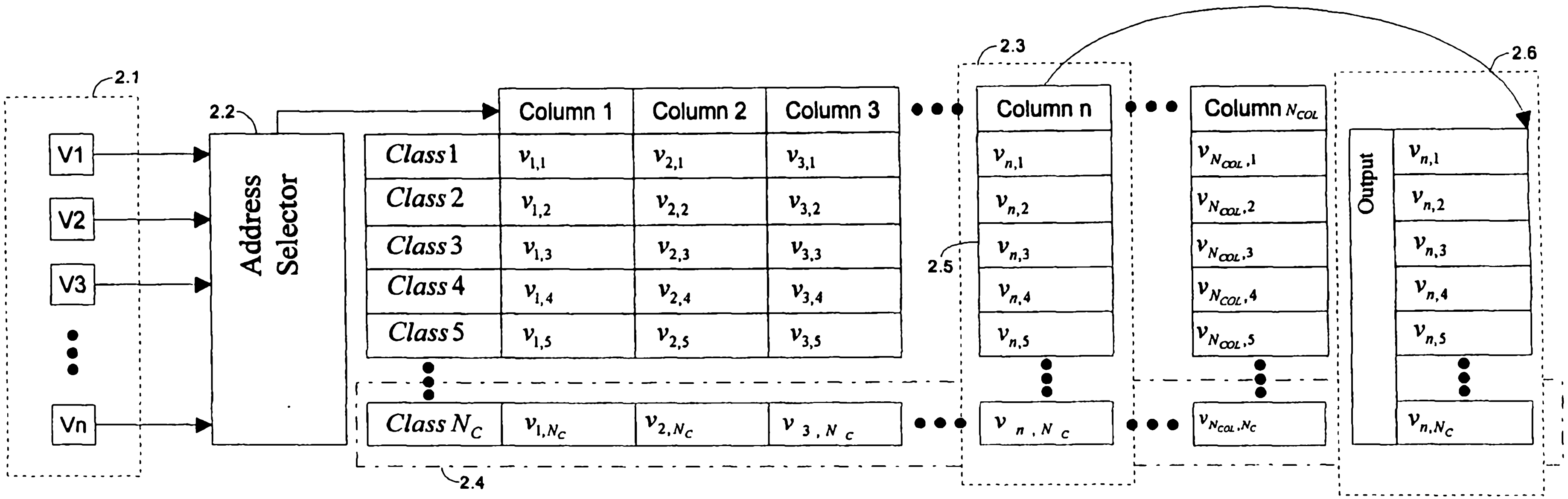

2.1
V1
V2
V3
Vn
2.2
Address Selector
Column 1
Column 2
Column 3
Column n
Column N_{COL}
2.3
2.6
Class 1
Class 2
Class 3
Class 4
Class 5
Class N_C
$v_{1,1}$ $v_{2,1}$ $v_{3,1}$ $v_{n,1}$ $v_{N_{COL},1}$
$v_{1,2}$ $v_{2,2}$ $v_{3,2}$ $v_{n,2}$ $v_{N_{COL},2}$
$v_{1,3}$ $v_{2,3}$ $v_{3,3}$ $v_{n,3}$ $v_{N_{COL},3}$
$v_{1,4}$ $v_{2,4}$ $v_{3,4}$ $v_{n,4}$ $v_{N_{COL},4}$
$v_{1,5}$ $v_{2,5}$ $v_{3,5}$ $v_{n,5}$ $v_{N_{COL},5}$
v_{1,N_C} v_{2,N_C} v_{3,N_C} v_{n,N_C} v_{N_{COL},N_C}
2.5
2.4
Output
$v_{n,1}$
$v_{n,2}$
$v_{n,3}$
$v_{n,4}$
$v_{n,5}$
v_{n,N_C}

Figure 3

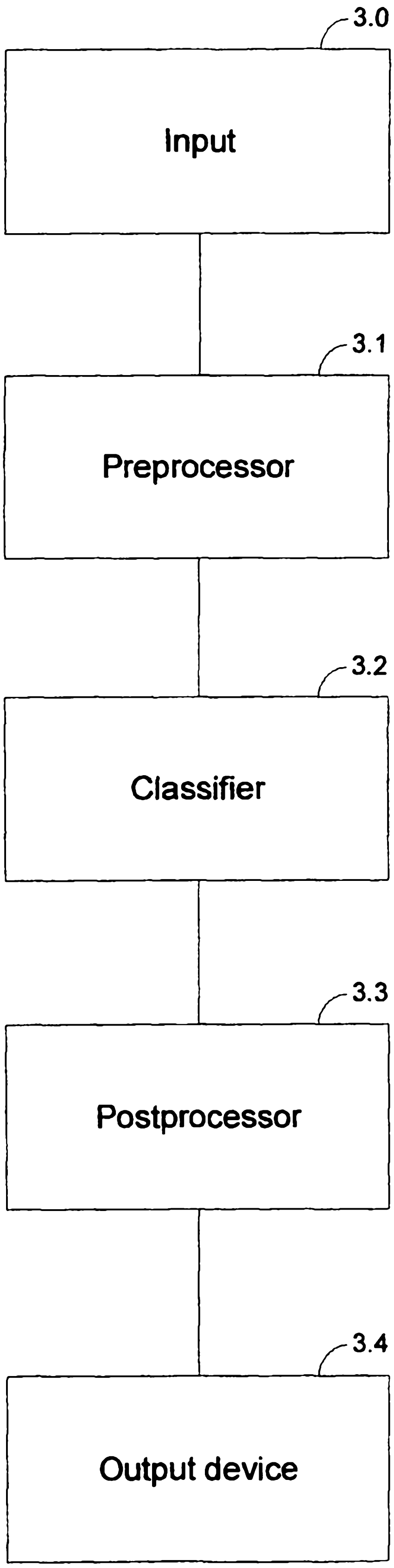

3.0
Input
3.1
Preprocessor
3.2
Classifier
3.3
Postprocessor
3.4
Output device

Figure 4

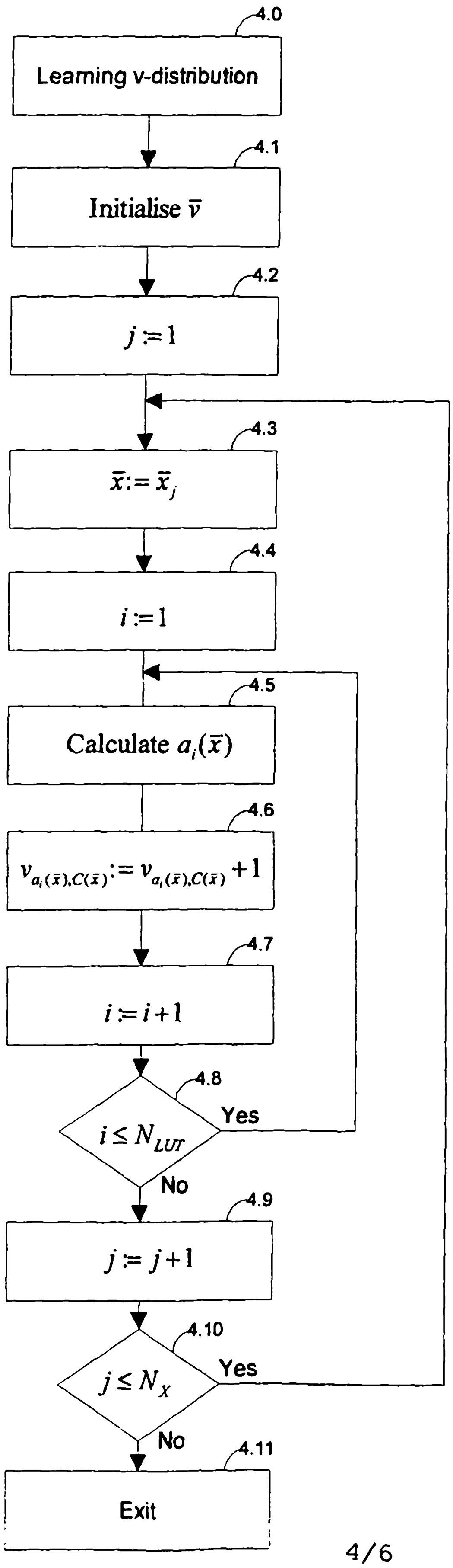

4.0
Learning v-distribution
4.1
Initialise $\bar{v}$
4.2
$j := 1$
4.3
$\bar{x} := \bar{x}_j$
4.4
$i := 1$
4.5
Calculate $a_i(\bar{x})$
4.6
$v_{a_i(\bar{x}),C(\bar{x})} := v_{a_i(\bar{x}),C(\bar{x})} + 1$
4.7
$i := i + 1$
4.8
$i \le N_{LUT}$
Yes
No
4.9
$j := j + 1$
4.10
$j \le N_X$
Yes
No
4.11
Exit

Figure 5

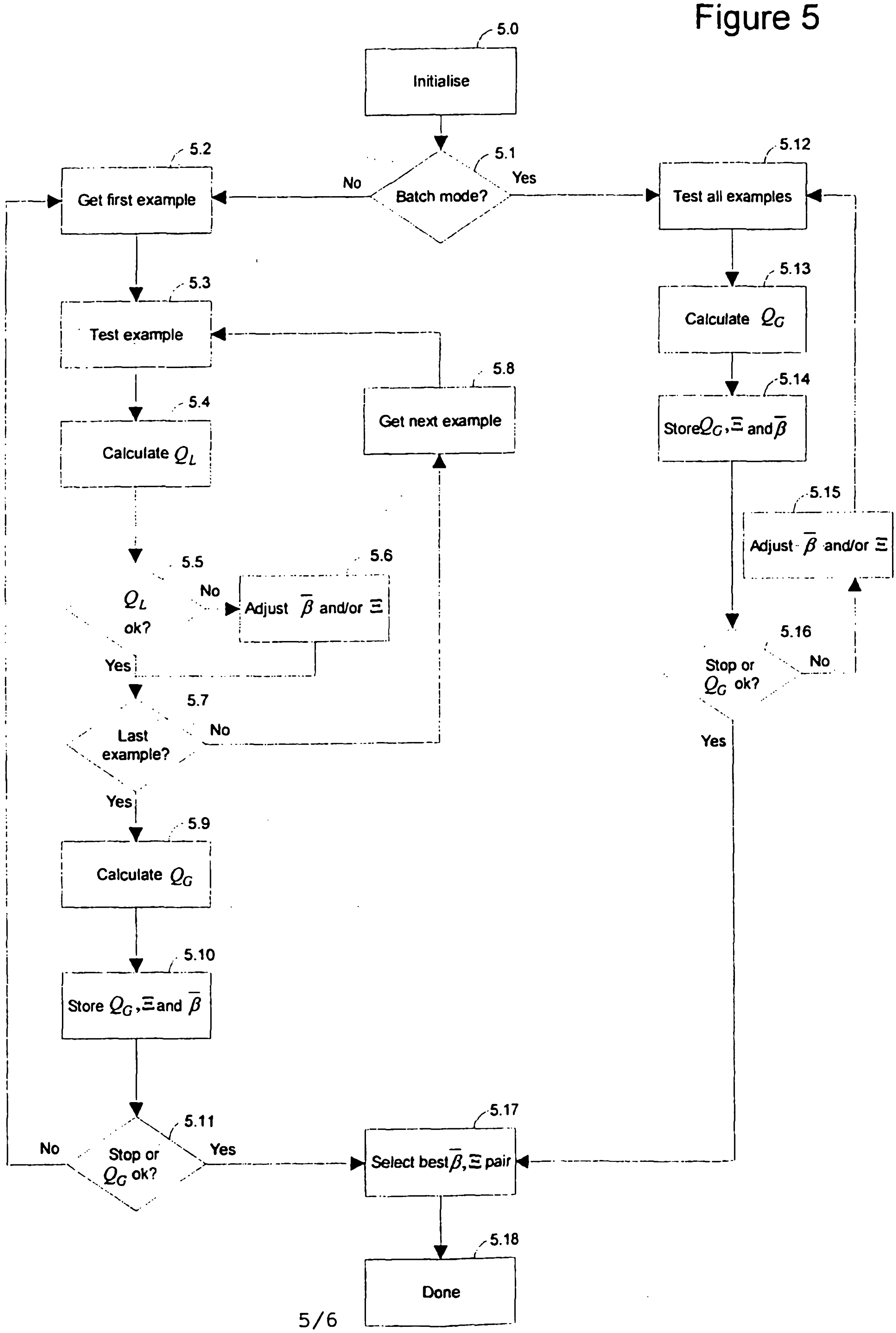
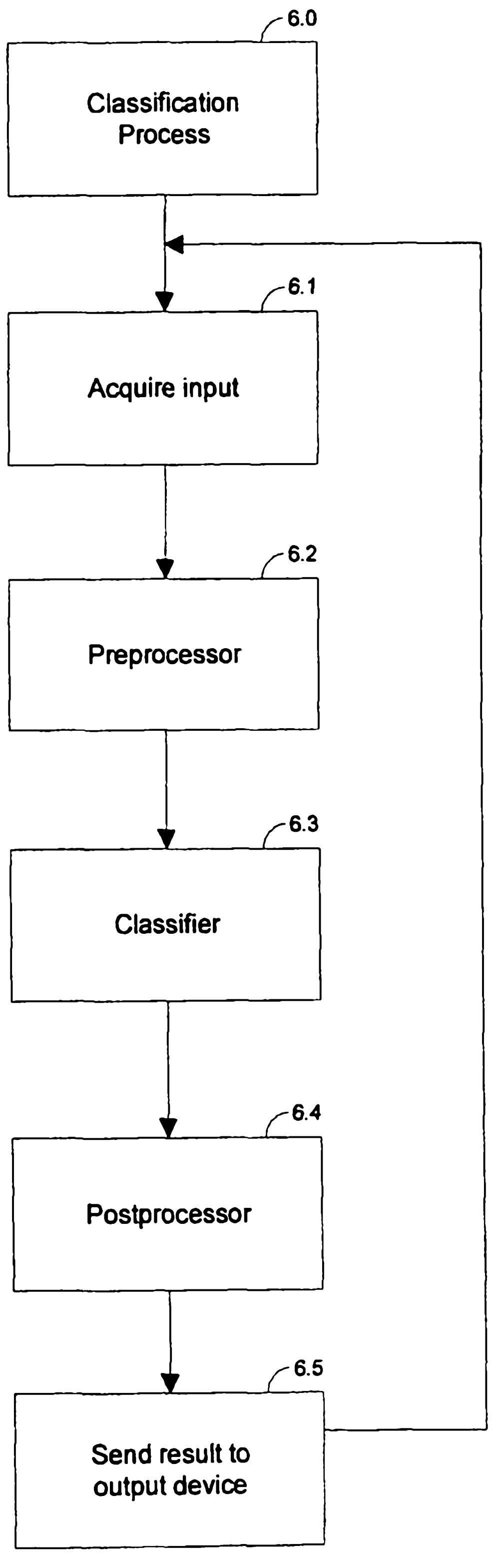

5.0
Initialise
5.1
Batch mode?
No
Yes
5.2
Get first example
5.3
Test example
5.4
Calculate Q_L
5.5
Q_L ok?
No
Yes
5.6
Adjust $\bar{\beta}$ and/or Ξ
5.7
Last example?
No
Yes
5.8
Get next example
5.9
Calculate Q_G
5.10
Store Q_G, Ξ and $\bar{\beta}$
5.11
Stop or Q_G ok?
No
Yes
5.12
Test all examples
5.13
Calculate Q_G
5.14
Store Q_G, Ξ and $\bar{\beta}$
5.15
Adjust $\bar{\beta}$ and/or Ξ
5.16
Stop or Q_G ok?
No
Yes
5.17
Select best $\bar{\beta}$, Ξ pair
5.18
Done

Figure 6


6.0
Classification Process
6.1
Acquire input
6.2
Preprocessor
6.3
Classifier
6.4
Postprocessor
6.5
Send result to output device