Johann Wolfgang Goethe-Universität
Frankfurt am Main

Fachbereich Informatik (20) Praktische Informatik *VSFT*

Proc. First Annual Meeting of the Internat. Neural Network Society, Boston 1988

PATTERN RECOGNITION AND FAULT TOLERANCE IN NON-LINEAR NEURAL NETWORKS

Dr. R. Brause
J.-W. Goethe University
FB 20, VSFT,Postfach 111932
D- 6000 Frankfurt 11, West-Germany

ABSTRACT:

It is widely agreed in the research of Neural Networks that non-linearities (i.g.thresholds) in the response of an activated neural element leads to a stable behavour when noise and crosstalk of other neuronal elements are suppressed.
The paper investigates this in further detail for a network where the input lines are coupled to the output lines by a connection matrix. This kind of device can be used as an associative memory [KOH1] if the matrix coefficients are learned by Hebb's law. By the introduction of a threshold to the linear output, the device output function becomes non-linear.
It is shown that the resulting device has pattern recognition and categorization properties. The classification of every input pattern to the most resembling stored input pattern can be interpreted as fault-tolerance for the input data.
In the paper three different threshold functions are investigated and the optimal thresholds are calculated. It is shown that the inherent fault tolerance proportions heavily depend on the coding (cross-correllation and distance) of the stored input patterns.

Furthermore, the hardware connection faults in a hardware model of the device are modelled as stuck_at_zero and stuck_at_one faults and the maximal occurence probabilities for the two kinds of faults are calculated for a (probably) correct operation of the device. It is shown that the hardware model is very insensitive for faults of open connections (depending on the coding of the stored patterns) but very sensitive for faulty, active connections which is an important fact in VLSI-implementations.

1. Introduction

... I don't think we ever debugged our machine completely, but that didn't matter. By having this crazy random design it was almost sure to work no matter how you built it.

Marvin Minsky about his learning machine

In modern parallel computer architectures a new generation of highly parallel, real-time oriented architecture for artificial intelligence is at the horizon. These attempts favorize computers made by many, small processing elements of very low complexety and therefore very limited computing power, connected directly together contrary to a relative small number of complex processors, communicating with a high amount of overhead. An example of these attempts is the connection machine [HILL].
One important class of highly functional parallel models are those proposed since 30 years by neurological and cybernetical scientists for modelling brain functions. The models are based on the function of simple elements, the neurons, connected extensively in a specific manner (*Neural Networks*). Every connection is assigned a specific weight.
These weights may represent special events or relations, therefore implementing directly semantic nets in hardware [FELD]. The connection models with dedicated connections have only a small degree of fault-tolerance because the failure of a node erases the whole associated event.
However, if the event is assigned to a specific state of the whole set of connections the inherent fault-tolerance properties are much more promising. Even in the early papers the surprising fault-tolerance, error-correcting and pattern completion properties are mentioned [WOOD],[KOH2], but never evaluated. Because the recall of the stored patterns are quite good, even when portions of the storage memory (weights) are erased, the patterns seem to be stored in a distributed manner like a holographic picture. The model was therefore termed *hologic memory* in the beginning [WILL2], [LONG].
Neural Networks can be used in a great variety of pattern processing tasks for high level AI functions, from adaptive filtering and feature extraction to decision making and storage of associative patterns [COMP]. For example, in many models of artificial intelligence the problem is divided into subproblems in a layered manner. In figure 1 the layers of computer vision and speech recognition systems are shown.

On each level the layer has to provide some basic fault-tolerant abilities like recognition of varied, noise-disturbed and incomplete patterns.

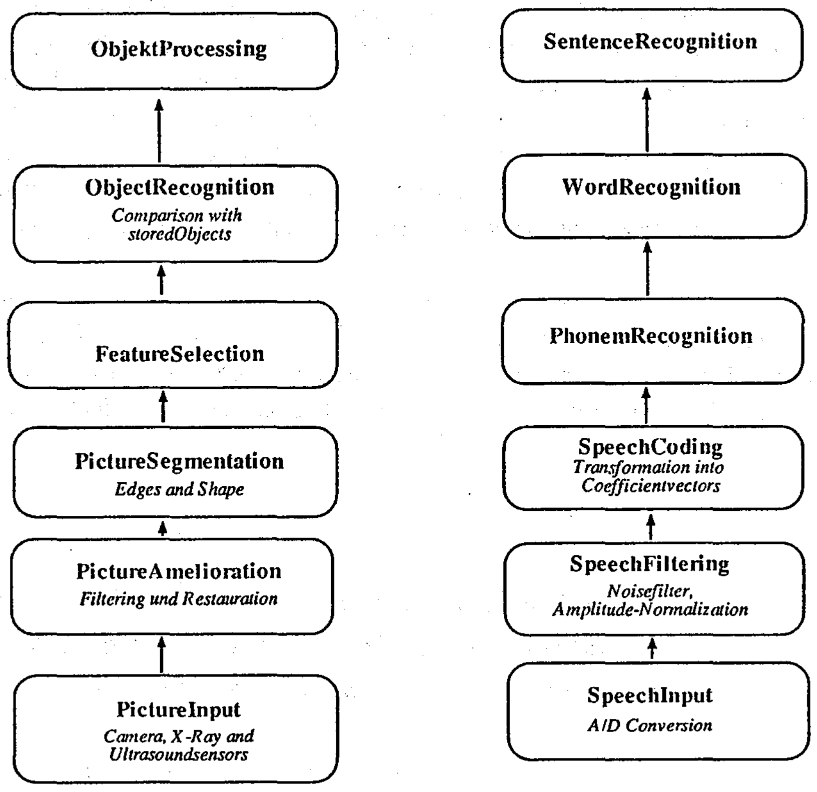


Fig 1 processing layers in computer vision and speech recognition

In the neural network approach each layer has the nearly the same homogen structure of interconnected, simple processing elements, e.g. [FUK].
In part 2 of this paper the network for one layer is introduced for use as an associative memory (see Kohonen [KOH1] and first McCulloch and Pitts [McCUL]) and some necessary conditions for pattern recognition and memory recall in the presence of disturbed input data will be given.
We regard here the basic configuration of a pure feedforward network without feedback cicuits. Contrary to the perceptron model, which also uses weights and a non-linear output transfer function, there is no training set of patterns for the associative memory; all patterns presented in the storage mode are completely stored.

In part 3 it is shown that the recall process of stored data can be viewed as a pattern recognition and error correction process which is controlled by a threshold. The optimal thresholds for two pattern similarity measures are evaluated and the optimal coding of input data is discussed.

A simple hardware model and its corresponding fault model is proposed in part 4 and the maximal number of connections whose failure or insufficient fabrication do not impede the proper recall process is derived.

Part 5 draws the conclusions of this paper.

2.0 The functional model

Let us first consider the formalization of the network concept.
Assume that we have *m* processing elements which have a processing function f(.), and *n* input lines which can be connected to the processing elements. The set of links (weights) between the input lines and the processing elements can then be described by a matrix $\mathbf{W} = (w_{ij})$. The output z_i of the interconnection network to the processing element i is a linear combination of the values of the input lines $\mathbf{x} = (x_1,...,x_n)^T$where T denotes the transpose. In a vector product $\mathbf{v}\mathbf{w}^T$the T will be omitted. The output y_i of the processing element i is therefore

$$y_i = f(z_i) = f(\sum_j w_{ij}x_j) \qquad \mathbf{z} = (z_1,...,z_m)^T \qquad (2.0a)$$

or $$\mathbf{y} = \mathbf{f}(\mathbf{z}) = \mathbf{f}(\mathbf{W}\ \mathbf{x})$$

This is the classical feedforward network as it is used in the perceptron model.
In Figure 2 the hardware model of the basic input-output configuration is shown. As you can notice, the fully connected network with its regular connections is fesible for VLSI implementations. The horizontal input lines can carry an input current on activation; by some resistor-like weights [STEIN] the current in the vertical output lines will be the weighted sum of the input activity.

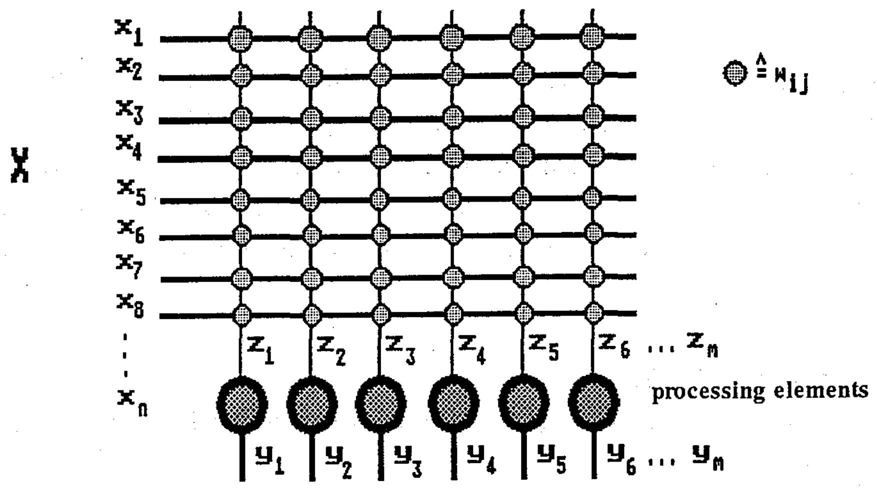

X
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_n
$\hat{=} w_{ij}$
z_1 z_2 z_3 z_4 z_5 z_6 ... z_m
processing elements
y_1 y_2 y_3 y_4 y_5 y_6 ... y_m

Fig. 2 Hardware model of the neural network

The network can be used as an associative memory in two different modes: the imprinting (storage) mode and the memory recall (readout) mode.

In the *storage mode* each time one of the sequentially presented input patterns $\mathbf{x}^1..\mathbf{x}^p$ (the class prototypes) with their associated output patterns $\mathbf{y}^1..\mathbf{y}^p$ appear, the weights are locally augmented by the correlation of input $\mathbf{x}^k$ and output data $\mathbf{y}^k$ with a proportional constant c_i.

$$\Delta w_{ij} = c_i^k\, y_i^k\, x_j^k \qquad \text{(Hebb's rule)} \qquad (2.0b)$$

After the presentation of p patterns the imprinting storage is complete:

$$w_{ij} = c_i^0 + \sum_{k=1}^{p} c_i^k y_i^k x_j^k$$

In the <u>*memory recall mode*</u> the recall is done by presenting an event to the input lines **x** and reading out the output at **y**

$$y_i = f\,(W\,\mathbf{x})_i = f\,(\sum_j w_{ij}x_j\,) = f\,(\,c_i^{\,0} + \sum_j \sum_k c_i^{\,k}\, y_i^{\,k} x_j^{\,k} x_j) = f(c_i^{\,0} + \sum_k c_i^{\,k} y_i^{\,k} \mathbf{x}^k \mathbf{x}) \qquad (2.0c)$$

It should be noted that the values of the weights are *not* the result of a stochastic approximation process as it is the case for instance in the perceptron algorithm but are determined deterministically.

2.1 The linear Projection Model

Let us consider the processing elements as simple analog amplifiers: $f(z_i) =: z_i$.
If we present a specific event $\mathbf{x}^r$ which was stored before, the recalled output is

$$y_i = z_i = \underbrace{y_i^{\,r}\, \mathbf{x}^r \mathbf{x}^r\, c_i^{\,r}}_{response} + \underbrace{\sum_{k,\, k\#r} c_i^{\,k}\, y_i^{\,k} \mathbf{x}^k \mathbf{x}^r}_{cross\text{-}talk} + c_i^{\,0} \qquad (2.1a)$$

It can be interpretated as the proper response $y_i^{\,r}$ and additionally evoked crosstalk from other stored patterns.
For stored events which are <u>orthogonally coded</u> the product $\mathbf{x}^k\mathbf{x}^r$ is 0. If we additionally normalize (2.1a) with the factor $c_i^{\,k} := (\mathbf{x}^k\mathbf{x}^k)^{-1}$ and $c_i^{\,0} = 0$ the equation (2.1a) becomes

$$z_i = y_i^{\,r} \mathbf{x}^r \mathbf{x}^r c_i^{\,r} = y_i^{\,r}$$

and the proper response $y_i^{\,r}$ is derived without any threshold involved. Since this memory model stores simply cross-correlations it was termed *correlation memory* [KOH1].
This is the model for which Kohonen demonstrated good pattern completion abilities [KOH2].
It should be noted that the linear model needs the orthogonalization of the input patterns to store them correctly. If a non-orthogonal (faulty) pattern is input, due to the linearity of (2.1a) the output will be faulty, too.

2.2 Orthogonal Projections

Since activity patterns of sensors of the real world do not provide orthogonal coded patterns generally, let us make two less restricting, but efficient assumptions:

1) Only the output is orthogonally coded:	$\mathbf{y}^k\mathbf{y}^r = 0$ for k#r
2) The activity (spike rate!) is only positive:	$x_i, y_i \geq 0$

From these two assumptions the sum (crosstalk) in (2.0d) will become zero; for every component i there exists at most one output pattern $\mathbf{y}^{k_i}$ which has the i-th component $y_i^{\,k_i}$ # 0.
So equation (2.0c) reduces to

$$y_i = f\,(\,c_i^{\,0} + c_i^{\,k_i} y_i^{\,k_i} \mathbf{x}^{k_i} \mathbf{x}\,) \qquad (2.2a)$$

In the case of the input of a "prototype pattern" $\mathbf{x}=\mathbf{x}^r$ and a zero component of the output pattern $y_i^{\,r} = 0$, k_i#r, a non-zero correlation $z_i = c_i^{\,0} + c_i^{\,k_i} y_i^{\,k_i} \mathbf{x}^{k_i} \mathbf{x}^r$ (the crosstalk) in the memory recall results in the output component. Certainly, if we normalize the weights and use only orthogonal input vectors (see above) we will get a proper recall.
Instead of using orthogonal input vectors, which is a strong restriction, and normalizing the weights let

us try to get the proper response by introducing a ***threshold function***

$$T(x) = \{ \begin{matrix} 0 & x \leq 0 \\ 1 & x > 0 \end{matrix} \qquad (2.2b)$$

which supresses the crosstalk in (2.2a).

2.3 Non-orthogonal Projections and Adaptive Thresholds

As a very simple form of a non-linear f(.) we can use the binary threshold function (see part 3.0)

$$\mathbf{y} = \mathbf{T}(\mathbf{z}\text{-}\mathbf{t}) \qquad \text{with } \mathbf{T}(\mathbf{v}) := (T(v_1), \dots, T(v_m))$$

With the addition of a constant component **x** becomes

$$\mathbf{x} = (x_1, \dots, x_n) \quad \mapsto \quad (x_1, \dots, x_n, 1)$$

the projection can be written in another way

$$\mathbf{y} = \mathbf{T}(\mathbf{z})$$

$$\text{with} \quad \mathbf{z} = \mathbf{W}\mathbf{x} \quad \text{and} \quad \mathbf{W} = \begin{pmatrix} w_{11}, \dots, w_{1n}, -t_n \\ \dots, \quad \dots, \quad \dots \\ w_{m1}, \dots, w_{mn}, -t_m \end{pmatrix}$$

The threshold level can therefore be set by some special "inhibitory connections", similar to the learning of the threshold in perceptron models.
This can be used for the case of non-orthogonal projections.
Let the class prototypes be chosen as to have a constant overlap $u := \mathbf{x}^k\mathbf{x}^r$ = const > 0. We can therefore formulate out classification or memory recall rule for an input of $\mathbf{x}^r$

$$\text{For } y_i^r = 0 \qquad z_i = (\mathbf{Wx})_i = \sum_k y_i^k \mathbf{x}^k \mathbf{x}^r = \sum_{k\#r} y_i^k u = u \sum_{k\#r} y_i^k := u\, s_i$$

$$\text{For } y_i^r = 1 \qquad z_i = (\mathbf{Wx})_i = \sum_k y_i^k \mathbf{x}^k \mathbf{x}^r = |\mathbf{x}^r|^2 + u\, s_i$$

In this case we can choose a threshold of $t_i := u\, s_i$ for correct memory recall.
If we treat the matrix coefficients $w_{i\,n+1} = -t_i$ as ordinary weights, we can manage the threshold adaption by an learning rule analogously to Hebb's law (2.0a)

$$\Delta w_{i\,n+1} = - u\, y_i^k$$

which results after storage completion in a matrix weight of

$$w_{i\,n+1} = - \sum_k y_i^k u = - u \sum_k y_i^k = - u\, s_i = -t_i$$

Therefore, by the introduction of a modified Hebb rule we can make the system learn the correct threshold for a non-orthogonal coding of input and output patterns.

For the rest of this paper let us now explore the orthogonal projection and its associated problem of the optimal threshold functions.

3.0 Non-linear Coupling and Pattern Recognition by Fault-tolerant memory Recall

There are many choices possible for the non-linear coupling function. Instead of using more complicated functions as for instance sigmoid functions (see e.g. [GROS]) let us regard the simple, non-linear threshold function T(x) of (2.0f). It has the advantage of an easy realization in modern digital computers. The different versions can be

$y = (z-t)\,T(z-t)$	suprathreshold linear	(3.0a)
$y = y\,T(z-t)$	real-valued threshold	(3.0b)
$y = T(z-t)$	binary threshold	(3.0c)

In figure 3.0a the three functions are shown.
Let us briefly regard the characteristics of the three models.

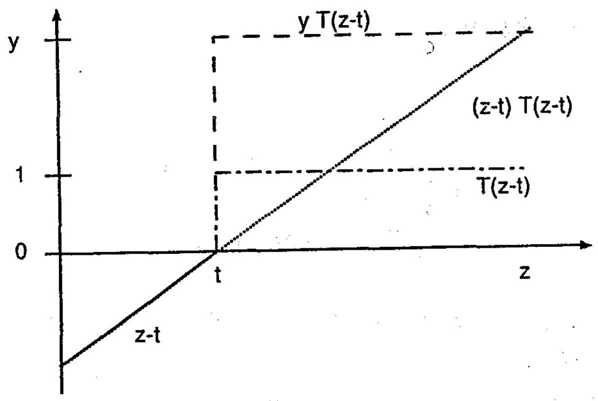


Fig. 3.0a The three different processing functions with thresholds

3.1 Suprathreshold linear

The function (3.0a) models the biological fact that the sum of the input activities must overrun a certain threshold before it causes an output activities. A class prototype can correctly be recalled only if

$$\text{for } y_i^r \# 0 \qquad y_i^r = z_i - t_i = y_i^r\,|\mathbf{x}^r|^2 - t_i^r \quad > 0 \qquad (3.1a)$$

$$\text{for } y_i^r = 0 \text{ and } y_i^{ki} \# 0 \qquad z_i - t_i = y_i^{ki}\,\mathbf{x}^{ki}\mathbf{x}^r - t_i^{ki} \quad \leq 0 \qquad (3.1b)$$

From (3.1a) we get directly the threshold

$$t_i^r = y_i^r\,(|\mathbf{x}^r|^2 - 1) \qquad (3.1c)$$

and from (3.1b)

$$\mathbf{x}^{ki}\mathbf{x}^r \leq |\mathbf{x}^{ki}|^2 - 1 \qquad k_i \# r \qquad (3.1d)$$

The relations (3.1c) and (3.1d) give us the threshold and a condition for the class prototypes. Besides, the conditions gives us some interesting fault-tolerance properties, too. Let us assume normalized activities, i.e. $|\mathbf{x}^{ki}|^2 =: a$ for all k_i. Then an arbitrary input pattern $\mathbf{x}$ is projected to

$$y_i^{ki} = (z_i - t_i)\,T(z_i - t_i) = (y_i^{ki}\,\mathbf{x}^{ki}\mathbf{x} - y_i^{ki}(a-1))\,T(z_i - t_i) = y_i^{ki}\,(\mathbf{x}^{ki}\mathbf{x} - (a-1))\,T(z_i - t_i)$$

With the definition

$$g(x^{ki},x) := x^{ki}x - (a-1)$$

all input patterns **x** which have $g(x^{ki},x) > 0$, which means a good correlation between x^{ki} and **x**, will cause an output greater zero.
As we can see in appendix D, if we consider normalized patterns with components of natural numbers the threshold condition (3.1c) and the condition of maximal correlation (3.1d) are necessary and sufficient conditions for consistent classification. As pattern **x** changes from x^r to x^k (r#k), we see that the function $g(x^{ki},x)$ will decrease from 1 to zero and eventually become negative ,cf (3.1d). Thus the response of the system to *faulty* versions of the stored prototypes will result in the *correct* output *with a lower amplitude*.
Let us now consider the general input-output behavour for arbitrary patterns **x**. From (3.0a) we get

for $y_i^r \# 0$ $\quad y_i^r = z_i - t_i^r = y_i^r xx^r - t_i^r > 0$ (3.1e)

for $y_i^r = 0$ and $y_i^{ki} \# 0$ $\quad z_i - t_i = y_i^{ki} xx^{ki} - t_i^{ki} \leq 0$ (3.1f)

With (3.1c) the two conditions become

for $y_i^r \# 0$ $\quad xx^r > |x^r|^2 - 1$ (3.1g)

for $y_i^r = 0$ and $y_i^{ki} \# 0$ $\quad xx^{ki} \leq |x^{ki}|^2 - 1$ (3.1h)

Let us illustrate the two restrictions for **x** by a two-dimensional plot.
The condition (3.1g) means

$$xx^r = |x^r|\,|x| \cos(x,x^r) > |x^r|^2 - 1$$

or with $|x| \cos(x,x^r) := \text{proj}(x,x^r)$ as *projection of x onto x^r*

$$\text{proj}(x,x^r) > |x^r| - |x^r|^{-1} = \text{const}$$

The second condition (3.1h) becomes for another stored pattern x^k

$$\text{proj}(x,x^k) \leq |x^k| - |x^k|^{-1} = \text{const}$$

These two restrictions are shown in figure 3.1a.
As we can see in the drawing 3.1a, the introduction of thresholds leads to the tolerance of small variations of the stored patterns x^r. A set of patterns, similar to the stored prototype, is projected on the same output pattern. This set of similar patterns can be regarded as a class of patterns, represented by the stored class prototype. Thus the memory recall becomes a pattern recognition, similar to that made by the perceptron model.
The mapping of a set of (real-valued) pattern vectors to one class prototype is also refered as *vector quantization.*
The class boarders are determined by the projection of **x** onto the class prototype vectors. The boarders are equivalent to thresholds and geometrically are hyperplanes orthogonal to the prototype vector in the distance $|x^k| - |x^k|^{-1}$.

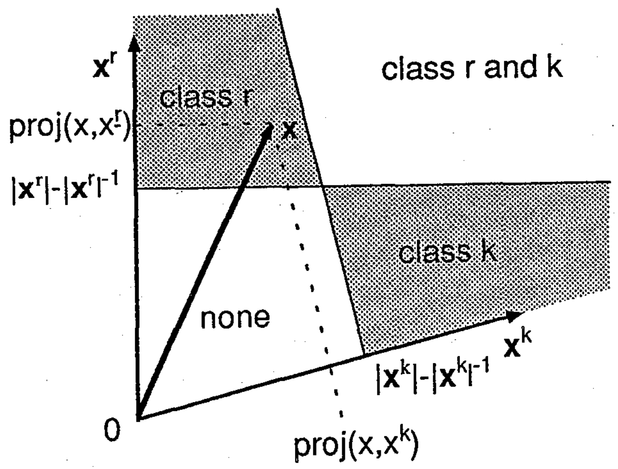


Fig. 3.1a classification by suprathreshold linear decision functions

Only those patterns **x** will be consistently classified which have projections to all prototype vectors (except just one) smaller than the class boundary hyperplanes.
If the length of the pattern vector is too big, the classification decisions will classify it in several classes at the same time, i.e. the resulting output pattern is a superposition of the appropriate class output patterns.
If the pattern is too small only the null vector results.

3.2 Real-valued and Binary Threshold

Let us first determine the threshold t_i^r which is sufficient for a proper recall of all class prototype vectors. For (3.0b) or (3.0c) we get the conditions

$$\begin{array}{lll} \text{For } y_i^r \# 0 & z_i(x^r) > t_i^r \\ \text{and } y_i^r = 0 & z_i(x^r) \le t_i^r \end{array} \qquad (3.2a)$$

These two conditions are necessary conditions, because they result directly from the definition (3.b). To be sufficient, the conditions must guarantee the proper recall of all class prototype patterns. This problem is transformed into the problem of finding a threshold with the desired properties. For the normalized threshold $t_i^r \mapsto t_i^r/y_i^r$ we get for both real-valued and binary thresholds with the "natural" choice $c_i^0{:=}0$ and $c_i^k{:=}1$ by (2.2a) and (3.2a) the condition

$$\max_{k,\, k\#r} \mathbf{x}^k\mathbf{x}^r \le t_i^r < \mathbf{x}^r\mathbf{x}^r = |\mathbf{x}^r|^2 \qquad (3.2b)$$

This relation guarantees us a suppression of the crosstalk and a proper memory recall for a prototype $\mathbf{x}^r$ by a processing unit i if the length of $\mathbf{x}^r$ (which is the number of ones in the binary case) is greater than the greatest overlap of $\mathbf{x}^r$ with another class prototype. This can be interpreted as a kind of *majority voting system* for fault suppression.
The condition (3.2b) defines only a valuable window for the t_i^r; it lets us some freedom for the choice of threshold. Let us try now to determine thresholds which implements a classification according to the rule "for a input **x** take the class with the most similar class prototype".
What does *most similar* mean exactly ?

Similarity Measures and Fault-tolerant Memory Recall

Let us look at some arbitrary input data pattern vector **x** which can be interpreted as a disturbed, faulty version of a class prototype $\mathbf{x}^r$. Our understanding of this fact is, that among all class prototypes $\mathbf{x}^k$ the input **x** mostly resembles to $\mathbf{x}^r$. The error correction of **x** and the recall of the pattern $\mathbf{y}^r$ becomes now an ordinary pattern recognition problem: we have to assign an unknown pattern **x** to the appropriate class which is represented by the class prototype $\mathbf{x}^r$. The classification rule "*take the most similar class prototype*" can be mathematically interpreted

1) by the demand for the *maximal cross-correlation*

$$\mathbf{x}\mathbf{x}^r = \max_k \mathbf{x}\mathbf{x}^k \qquad (3.2c)$$

or

2) by the demand for the *minimal distance*

$$|\mathbf{x}-\mathbf{x}^r| = \min_k |\mathbf{x}-\mathbf{x}^k| \qquad (3.2d)$$

Certainly, the two measures are coupled : $|\mathbf{x}-\mathbf{x}^r|^2 = |\mathbf{x}|^2-2\mathbf{x}\mathbf{x}^r+|\mathbf{x}^r|^2$

Let us now try to understand the different meanings of the two measures for our problem by a geometric interpretation.

Geometrical Interpretation

The demand for maximal cross-correlation means that we choose as classprototye for **x** the pattern $\mathbf{x}^r$ which satisfy

$$\mathbf{x}\mathbf{x}^r > \mathbf{x}\mathbf{x}^k \qquad \text{for every class } k\#r \qquad (3.2e)$$

The boundary between two classes $\mathbf{x}^r$ and $\mathbf{x}^k$ is given with $\{\mathbf{x}^* | \mathbf{x}^*\mathbf{x}^r = \mathbf{x}^*\mathbf{x}^k\}$.

With the distance $\mathbf{d}^{rk} := \mathbf{x}^r-\mathbf{x}^k$ the equation $\mathbf{x}^*(\mathbf{x}^r-\mathbf{x}^k) = 0$ of the boundary becomes $\mathbf{x}^*\mathbf{d}^{rk}=0$ and the boundary $\{\mathbf{x}^* | \mathbf{x}^*\mathbf{d}^{rk}=0\}$ consists of the hyperplane which is orthogonal to the distance vector $\mathbf{d}^{rk}$ between the two class prototypes.

In figure 3.2a the situation is illustrated.

This boundary means for all **x** on the right hand side of the plane, that the angle α between **x** and $\mathbf{d}^{rk}$ is $-90^0 < \alpha < +90^0$ and so $\cos(\alpha) > 0$.

Thus

$$0 < \cos(\alpha)\, |\mathbf{x}||\mathbf{d}^{rk}| = \mathbf{x}\,\mathbf{d}^{rk} = \mathbf{x}\,(\mathbf{x}^r-\mathbf{x}^k)$$

or $\mathbf{x}\mathbf{x}^r > \mathbf{x}\mathbf{x}^k$ which is the condition (3.2a) for maximal cross-correlation.

As we can see, the classification works quite good in our illustration. What are the problems of this classification scheme ?

For a correct classification of $\mathbf{x}=\mathbf{x}^r$ into the class r the relation $\mathbf{x}^r\mathbf{x}^r > \mathbf{x}^r\mathbf{x}^k$ must hold. Therefore, our whole pattern space is devided again in two sets by a hyperplane with

$$\{\mathbf{x}^* | \mathbf{x}^r\ (\mathbf{x}^r-\mathbf{x}^*) = 0\}$$

i.e. the difference vector $\mathbf{d}^{rk} := \mathbf{x}^r-\mathbf{x}^k$ is orthogonal to $\mathbf{x}^r$.

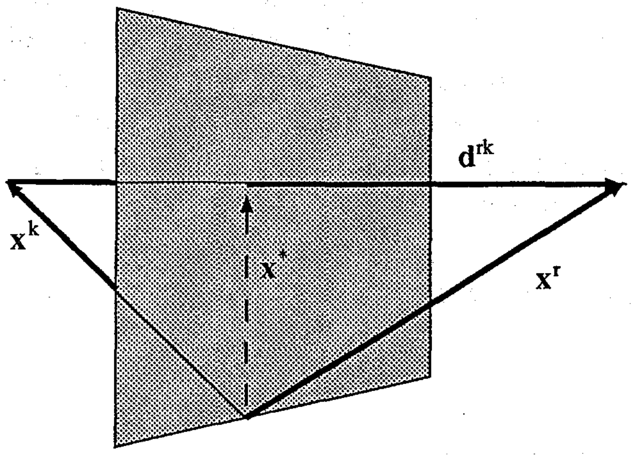


Fig.3.2a The class boundary between two classes for n=3

To allow a correct classification of the prototype vector itself all other prototype vectors should not be in the area bounded by the hyperplane orthogonal to $\mathbf{x}^r$. In figure 3.2b the "forbidden areas" are shown gray-shaded for three class prototypes in the 2-dim case.

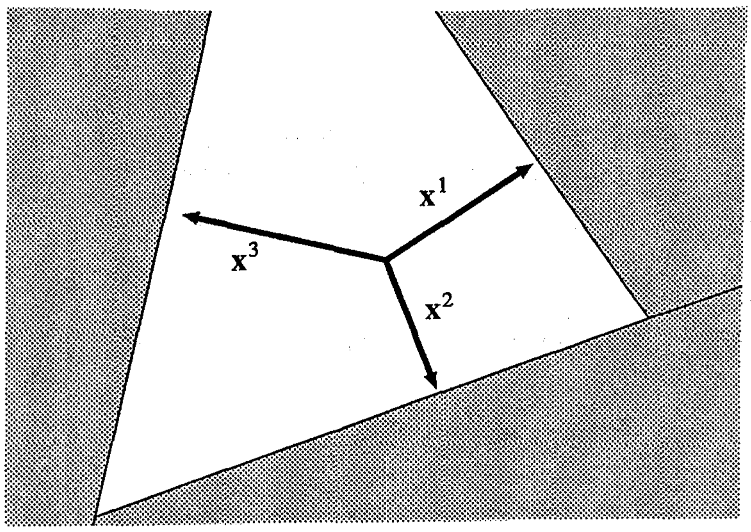


Fig.3.2b restrictions for correct recognition

Let us now look at the other similarity measure, the distance to the class prototypes.
The classification schema (3.2a) is eqivalent with a tesselation of the pattern space; the boundary between two classes is a hyperplane which intersects orthogonally the difference vector $\mathbf{d}^{rk}$ at $\mathbf{d}^{rk}/2$, see figure 3.2c. The proof is in appendix A.
It should be noted that the set of class prototype vectors in figure 3.2c cannot correctly be recognized by the classification rule (3.2c).

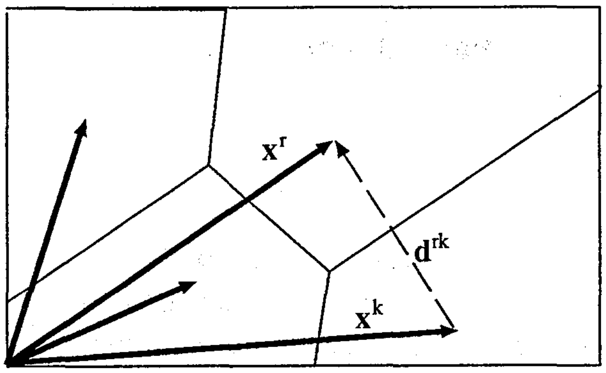


Fig.3.2c tesselation of the pattern space

It is interesting to note that the set of class protypes can be seen as a state in Kohonens *topology conserving mapping* algorithm (see [KOH3]). This can be used for instance to implement the optimal mapping by choosing the appropriate class prototypes (e.g. a set of equally-spaced vectors). As we can see in figure 3.2d, the mapping can only approximately implemented by the associative memory device.

Optimal Thresholds and Interprocessor Communication

As we have seen in the previous geometrical interpretations, the classifications are determined by the class boundaries. The two classification rules can be implemented as threshold decisions; e.g. with the threshold of (3.2a) every processing element compute whether the activation is strong enough ($t_i^r(k) < z_i$) to belong to class r or is just crosstalk ($t_i^r(k) \geq z_i$) due to class k.

Two problems arise. *First,* it is only valid for two classes r and k. If we have more classes, we will have more possible thresholds; to distinguish between more similar protypes the border demands higher correlations. For the necessary threshold decision we have to know all the other correlations obtained at the other processing elements as it is stated in the classification rule of (3.2c).

This is the *second* problem: the proposed parallel network model does not contain communication between the processing elements.

As solution to this problem we have to choose a threshold t_i^r, which do not depend on the other classes k, thus preventing the communication.

On the one hand we have to consider the worst case and choose the highest of all thresholds to guarantee the correct class-membership of the recognized patterns, our pattern recognition process will assign on the other hand a certain number of patterns of class r to the null vector class and recognize only the most similar ones.

As we can see, in this model without communication (e.g. "lateral inhibition" or other feedback) we can not implement the best possible classification of (3.2c) or (3.2d) but only a sufficient one.

Maximal Crosscorrelation

The sufficient threshold condition for an arbitrary pattern **x** has to be the same as in (3.2b)

$$\max_{k,\ k\#r} \mathbf{x}\mathbf{x}^k \leq t_i^r < \mathbf{x}\mathbf{x}^r \qquad (3.2f)$$

With some geometrical considerations (see appendix B) we get the threshold between class r and k.

With $K^r := \max_k x^k x^r$ and normal activity $|x^r|^2 = |x^k|^2 =: a$

we get by (B.4) from appendix B

$$t_i^r = |x| \left(1/2(a+K^r) \right)^{1/2} \tag{3.2g}$$

<u>Minimal Distance</u>

For the general case the optimal threshold is calculated in appendix C.

Let us now regard the interesting <u>binary case</u>. The cross-correlation gives for two binary vectors **v** and **w** the number of common components having the value '1'. Let us now consider another measure of similarity: the **Hamming distance** $d_H(\mathbf{v},\mathbf{w})$, defined as the number of components which are different between the two vectors **v** and **w**. What relations hold between the two measures?

The number of non-zero components in the distance vector (**v**-**w**) is just the number of components which are different between the two binary vectors. Since the number of '1' in a binary vector **x** is $|x|^2$, the quadratic Euklidean distance for binary vectors is the Hamming distance:

$$d_H(\mathbf{v},\mathbf{w}) = d_E(\mathbf{v},\mathbf{w})^2 = (\mathbf{v}-\mathbf{w})^2 = \mathbf{v}^2 - 2\mathbf{v}\mathbf{w} + \mathbf{w}^2 \tag{3.2h}$$

Further calculations (see appendix C) gives us for the binary case the sufficient threshold with the minimal Hamming distance $d_H^r := \min d_H(\mathbf{x}^r,\mathbf{x}^k)$

$$t_i^r := 1/2\,(|x^r|^2 + |x|^2 - d_H^r/2) \tag{3.2i}$$

As we can see, a good threshold is determined by the shortest distance of $\mathbf{x}^r$ to its class boarder. This results in the classification strategy of assigning only those **x** to class r which are in a secure neighbourhood of $\mathbf{x}^r$. The boarder of these neighbourhoods correspond to the circles in figure 3.2d. Certainly, there are many patterns which are in no circle and are therefore projected to the null vector. We can compensate this effect by a sufficient enlargement of the circles, eleminating the space between the circles and the boundaries. The resulting mapping of the input patterns is no more an exact orthogonal projection but only an approximate one. The recalled output patterns **y** of patterns **x** belonging to class *r* will be very close (very short distance) to the output pattern $\mathbf{y}^r$ associated to the class prototype $\mathbf{x}^r$, but not necessary the same.

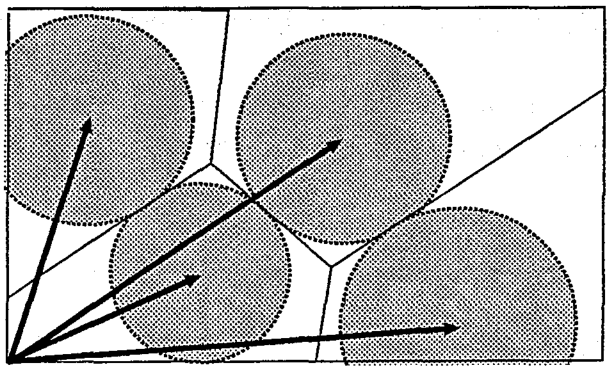


Fig. 3.2d Classification regions

Both thresholds (3.2d) and (3.2i) use the pattern strength $|x|^2$ to set up the threshold. This can be accomplished by a simple hardware addition to our hardware model of figure 2. This is shown in figure 3.2e .
The whole device can be implemented as a VLSI-chip of very regular structures. The summation for every output component y_i can be easily done by the superposition of the currents caused by different input lines. The connections are then realized as diode/resistor combinations. The modifiable resistor can be a physical device like a EEPROM connection [GOS] or a binary counter. If the resulting current in the column is greater than the threhold value t_i, which in turn is set by some internal constants and the external (see square elements in figure 3.2e) generated $x=|x|^2$; the threshold element sets the output y_i from 0 to 1. This is done immediately, so the whole search and pattern recognition process takes only one cycle which is with the technology of today in the range of nanoseconds.

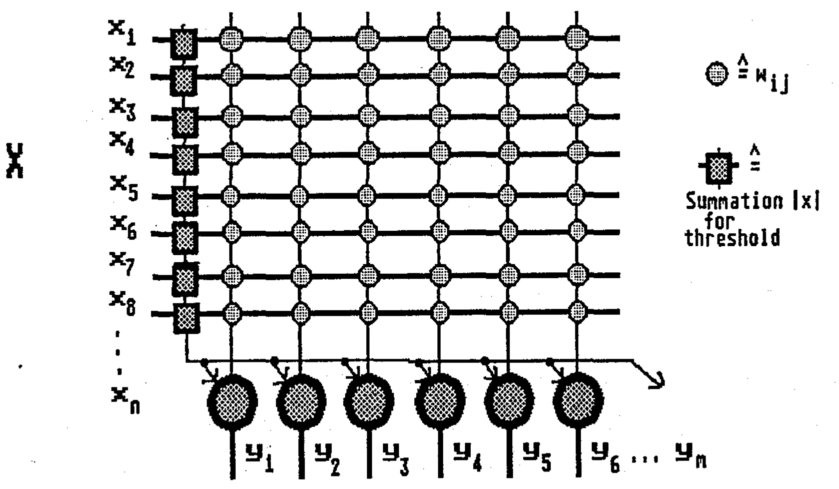

X
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x_n
$\hat{=} w_{ij}$
$\hat{=}$
Summation |x| for threshold
y_1 y_2 y_3 y_4 y_5 y_6 ... y_n

Fig. 3.2e Modified hardware model for binary patterns

It should be noted that in the binary case we can get another threshold without the need for changing our hardware model of figure 2. The threshold relation of the class-boarder $d_H(x,x^r) < d_H^r/2$ (see appendix C) can be expressed as

$$|x^r|^2 - d_H^r/2 < 2xx^r - |x|^2$$

Instead of changing the hardware structure we may only change the constants in the Hebbian storage rule we use. With c^0:=-1 and c_i^k:=2 we get from equation (2.2a) $z_i = 2xx^r - \Sigma_j x_j$ which is in the binary case $z_i = 2xx^r - |x|^2$. According to the threshold conditions (3.2a) the threshold can therefore be chosen as

$$t_i^r := |x^r|^2 - d_H^r/2 \qquad (3.2j)$$

In the binary case at normal activity a:= $|x^r|^2$ the optimal threshold is determined by the minimal Hamming distance between the stored input pattern class prototypes. If all class prototypes have the same Hamming distance d then the decision boarder between two classes is given by d/2.
It is interesting that the above classification rule (3.2j) coincidences well with the result of coding theory,

which states that error-correction in block codes can only be obtained if the disturbed codeword has a Hamming distance lower than half of the minimal Hamming distance between two codewords.

3.3 Fault Tolerance, Pattern Completion and Relational Database

Let us now consider a special case of fault tolerance in the memory recall: the pattern completion operation.
Pattern completion is obtained when one of the class prototype pattern vectors is only partially filled. The sparse vector is treated like any other faulty input data: by the threshold mechanism it is mapped into an appropriate class. If the input and output coding are the same, the classification and fault-correction of the incomplete input data results in the output of the completed input pattern.
This can be seen as a very fast request to a relational data bank. For example, a relational tuple *(relation, objectl, object2)* can be coded by the concatenation of the codes for relation, *objectl* and object2, cf.[HIN2]. The resulting long vector can be stored in the memory, associated with itself so that $\mathbf{y}^k=\mathbf{x}^k$. If we have only the incomplete tuple, for instance *(relation, objectl, -)* and we are searching for the complete one, all we have to do is to present the incomplete $\mathbf{x}^k$ (which have some '1' lacking) as an input pattern to the memory device. If the tuple was properly coded and the Hamming distance to the other stored tuples is big enough, then the complete relation will be output. Thus the basic functional proportions and the fault-tolerance abilities are tied intrinsically together.

3.4 Optimal Coding of Input Data

In most of the papers dealing with associative memory, the coding of the input and output vectors are not treated, in spite of the fact that the memory recall is very sensible to the overlap, i.e. to the Hamming distance of the stored patterns. For the optimal fault-tolerant, error-correcting memory recall in a real implementation of an associative memory it is important to obtain some guide lines for the optimal coding of the input events which yields maximal functionality and error-correction. Since the optimal coding is dependant on the requirements of the input data, two different attempts are presented.

a) Suppose, we want the maximal possible fault-tolerance. This is obtained by the maximal possible Hamming distance d.

$$\max_{r,k}\ d(\mathbf{x}^r,\mathbf{x}^k) = \max_{r,k}\ |\mathbf{x}^r|^2 + |\mathbf{x}^k|^2 - 2\mathbf{x}^r\mathbf{x}^k = 2a \qquad\qquad |\mathbf{x}^r|^2 = |\mathbf{x}^k|^2 =: a$$

This is obtained for $\mathbf{x}^r\mathbf{x}^k=0$, i.e. all class prototypes are orthogonal. The number N(a) of possible prototypes is then quite small:

$$N(a) = \lfloor n/a \rfloor$$

Example : With n=10 and a=3 we have d=6 and only N(a) = 3 class prototypes.

b) Suppose, we want as many events coded randomly with $|\mathbf{x}^k|^2=a$ as possible and want to have the maximal expected Hamming distance *d* between the events.
What is the optimal *a* ?

$$E(d(\mathbf{x}^r,\mathbf{x}^k)) = 2a - 2E(\mathbf{x}^r\mathbf{x}^k) = 2a - \sum_{i=1}^{n} E(x_i^r)E(x_i^r)$$

with the expectation function E(x).

With $E(x_i) = 0\,P(x_i=0) + 1\,P(x_i=1) = a/n$

we have $E(d(\mathbf{x}^r,\mathbf{x}^k)) = E(d(a)) = 2a - 2n\,a/n\,a/n = 2(a - a^2/n)$

The expectation value is maximized at a*

$$\frac{\partial}{\partial a} E(d(a)) \Big|_{a=a^*} = 2(1-2a^*/n) = 0$$

and therefore the optimal length or "activity" of a prototype vector is $a^* = n/2$ with the maximal expected Hamming distance $d = n/2$.

The number of possible different prototypes is

$$N(a^*) = \binom{n}{a^*} = \binom{n}{n/2}$$

It is interesting to consider the question

What is the value of a *which maximizes the number* N *of possible prototypes ?*

It is

$$N(a) = \binom{n}{a} = \binom{n}{n-a} = \binom{n}{s} \quad \text{with } s := n-a$$

Since N(a) is monotonically increasing with a = 1, 2, ... a<<n, and this goes also with increasing s (i.e. decreasing a) for a = n, n-1, ... the function has a maximum at a=s and therefore

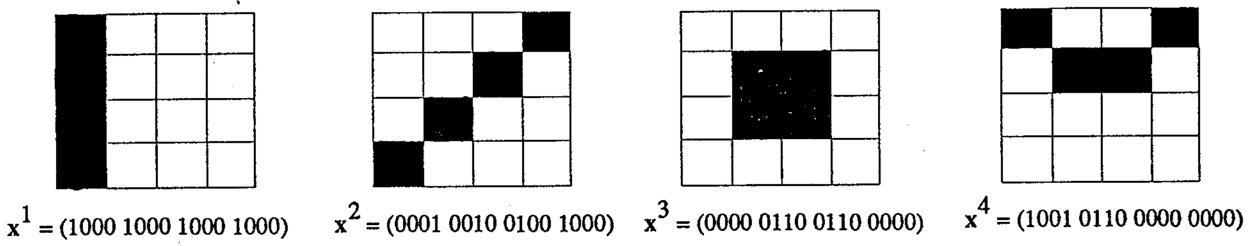
$$a^* = s^* = n-a^* \quad \Rightarrow \quad a^* = n/2$$

The optimal length of a vector with the maximal expected Hamming distance yields also the maximal number of possible vectors.

Example : For n=10 we have a*=5, d=5 and N(a*)=252 different prototype patterns.

3.5 Example

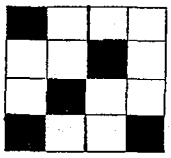
Let us illustrate the function principles of the non-linear model by the example of an auto-associative memory. We have four different class prototypes of dimension n=16. This can be visualized by a picture, composed of 4x4 pixels. When a pixel p_{ij} is black, the corresponding component 4i+j in **x** is '1'.
In figure 3.5a the prototypes are shown.



x^1 = (1000 1000 1000 1000)
x^2 = (0001 0010 0100 1000)
x^3 = (0000 0110 0110 0000)
x^4 = (1001 0110 0000 0000)

Fig.3.5a visualized prototype vectors of the example

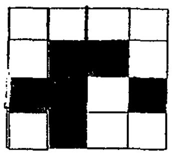
Let us assume that the prototype vectors are stored (associated with itself) and the thresholds $t = 1+|\mathbf{x}|^2/2$ are set up. Now we input three different patterns $\mathbf{x}$: first, an incomplete version of $\mathbf{x}^2$, then a noisy version of $\mathbf{x}^3$ and finally a pattern which has the same (minimal) distance both to $\mathbf{x}^2$ and $\mathbf{x}^3$. The input patterns and their output results are shown in the following figure 3.5b.



Input: $x = (1000\ 0010\ 0100\ 1001)$ $t = 3.5$ $d(x,x^1) = 5$ $d(x,x^3) = 5$

Output: $y = x^2$ $d(x,x^2) = 3$ $d(x,x^4) = 5$

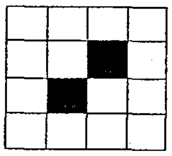
<u>**completion operation**</u>



Input: $x = (0000\ 0110\ 1101\ 0100)$ $t = 4$ $d(x,x^1) = 8$ $d(x,x^3) = 4$

Output: $y = x^3$ $d(x,x^2) = 6$ $d(x,x^4) = 6$

<u>**noise reduction operation**</u>



Input: $x = (1000\ 0010\ 0100\ 1001)$ $t = 3.5$ $d(x,x^1) = 6$ $d(x,x^3) = 2$

Output: $y = x^0 = (0000\ 0000\ 0000\ 0000)$ $d(x,x^2) = 2$ $d(x,x^4) = 4$

<u>**not decidable**</u>

Fig. 3.5b input patterns and the associated output

It should be noted that this example can not be used for visual pattern recognition because the pattern similarity measures which are used here do not correspond to visual similarity. For instance, if the vertical bar of $\mathbf{x}^1$ is shifted one pixel it is visually similar, but the resulting pattern vector is orthogonal to $\mathbf{x}^1$.

4.0 Hardware Fault Tolerance

In the former part of this paper we took a closer look to the fault-tolerance properties in the processing of input data. Beside this functional aspect we want to know now: what is the hardware fault tolerance, typical for this kind of design? To what extent of degradation does the device continue to function properly ?
In comparison with its biological counterparts the matrix model with its complete connected units has too much connections. Are they all necessary? Under what circumstances do the device continue to function, even in the presence of failures or lacking connections?

4.1 The Linear Model

As we can see in part 2.0 the input of faulty data yields faulty output, too. Certainly, this is also true when we apply valid data to a matrix of randomly failed connections. Kohonen calculated in [KOH1] the mean and variance of the output patterns. He also showed in [KOH3], p.165, that in the case of orthogonal output patterns (orthogonal projection) an uniformly distributed random error $e_x := |x-x^r|$ in the input data is attenuated to the projection $e_y := |y-y^r|$ of the output by

$$\mathrm{var}(e_y) = p/m\ e_x^2$$

Clearly, when the number of classes p is smaller than m= dim(y), the noise is diminuished.
By a memory matrix with failed connections. pattern recognition can be successfuly made if only the maximal component is taken.
Even for the operation of the linear model Kohonen found [KOH3,p114] that not all connections must be made; a relation of 40 between the number of input lines and the number of connections should be sufficient. Neither this nor other fault-tolerant statements [WOOD] are justified analytically.
Let us do this now for our threshold model of section 3.2.

4.2 The Fault Model

As a hardware model let us assume the functional model of figure 3.2c with the threshold function of (3.2i). To explore the maximal fault-tolerance capabilities of our model let us assume that all class prototypes $\mathbf{x}^1 ... \mathbf{x}^p$ are maximal fault-tolerant coded, i.e. the $\mathbf{x}^k$ are orthogonal (cf. section 3.4a). Since the output y^k is orthogonal, too, the weights are reduced for $y_i^k=1$

$$w_{ij} = \sum_k y_i^k x_j^k = \sum_k x_j^k = x_j^k$$

So the weights can only have the two values, 0 or 1.
Then, for the ease of the model, let us consider *only two kind of resulting faults* by defective hardware elements: stuck_at_one and stuck_at_zero. This is one of the most simple assumption which are possible, but it will already show us some interesting fault tolerance properties of the model. More complicated fault models should be set up with a concrete hardware implementation on hand.
The components which can be faulty are the connection elements, the threshold elements and the summation elements (square elements in figure 3.2e) of $|x|^2$. If we have for example 1000 input lines and 1000 output lines the number of connections are 10^6. In this example the threshold and summation elements constitute (with the same hardware complexity) only 0.2% of the hardware elements. If they fail, the output will be erroneous, of course, and must be corrected by the next matrix device.

The main problem is the amount of connections: *what will be if they fail?*
For the failure of connections w_{ij} with stuck_at_1 we must distinguish two kinds of failures: **active failures** which produce constantly a '1' (e.g. binary counters) and **passive failures** which will cause a '1' only if the corresponding input line is activated (e.g. EEPROM connections).
In a large number of hardware independent connections we can neither assume that all faults are on the same input line (which will just cause an input error of one bit) nor that they are all on the same output line (which will be corrected in the following layer). Instead we will assume in the following evaluation that the faults are equally distributed in the whole connection matrix.

4.3 Tolerable Hardware and Input Faults

Let us denote the failure probabilities

P_0 := P(connection defect and stuck_at_0)
P_1 := P(connection defect and stuck_at_1)

and assume that the faults occur independently.

Our question of 4.0 in this context is now:
How many connections can fail without producing erroneous output when a input pattern is presented?

When a pattern of class r is applied the erroneous activity results in an erroneous z_i:

$$z_i \rightarrow \text{error}(z_i)$$

and we have two situations where a faulty output is produced:

1) A column signal y_i^r is turned from 0 to 1 if too many connection weights are stuck_at_1. With (2.2b) this is equivalent to

$$\text{error}(z_i(\mathbf{x})) > t \qquad \text{with } \mathbf{x} \text{ of class } k \# r$$

2) A column signal y_i^r which should be 1 is turned to 0 if too many connections are stuck_at_0. With (2.2b) this means

$$\text{error}(z_i(\mathbf{x})) \leq t \qquad \text{with } \mathbf{x} \text{ of class } r$$

Let N_0 denote the activity (number of ones) originally induced by the input pattern, suppressed by connections stuck_at_0 and therefore subtracted from the original activity z_i in column i.
Additionally, let N_1 denote the activity which is erroneously added to the output z_i of column i and which was produced by connections stuck_at_1 in column i.
No error will occur and the pattern **x** will be consistently classified to class r and invoke a correct output pattern if the inverse of the two error conditions hold

$$\begin{array}{llll} \text{For} & y_i^k{=}1 & \text{error}(z_i(\mathbf{x})) = z_i(\mathbf{x})+N_1-N_0 := S \leq t^k & k \# r \qquad (4.3a) \\ \text{and} & y_i^r{=}1 & \text{error}(z_i(\mathbf{x})) = z_i(\mathbf{x})+N_1-N_0 = S > t^r & \end{array}$$

Let us now evaluate the numbers N_0 and N_1.

By definition, N_0 is the number of connections stuck_at_0 which have the weight w_{ij} # 0 and $x_j = 1$. Since we have $w_{ij} = x_j^r$ in the i-th column we get with appendix E

$$N_0 = \mathbf{xx}^r P_0 = 1/2\ (|\mathbf{x}|^2 + a - d(\mathbf{x},\mathbf{x}^r))\ P_0$$

For the calculation of N_1 we have to distinguish between the active and passive failure model.

<u>**Active fault model**</u>

In the *active fault model* beside the $\mathbf{xx}^r$ activated connections we have n - $\mathbf{xx}^r$ connections in column i where a stuck_at_1 will produce additional activity. With appendix E we have out of (n - $\mathbf{xx}^r$) connections with failure probability P_1 the activity

$$N_1 = (n - \mathbf{xx}^r)\ P_1$$

The sum S in the two conditions (4.3a) for correct memory recall becomes

$$S_a = z_i - N_0 + N_1 = \mathbf{xx}^r (1-P_0-P_1) + n\ P_1$$

With the Hamming distance (3.2h) and normal activity $|\mathbf{x}^r|^2 = |\mathbf{x}^k|^2 = a$ and the minimal Hamming distance of $d_H^{\ r} = d_H^{\ k} = 2a$ (orthogonal prototypes) we get

$$t^r = 1/2\ (a + |\mathbf{x}|^2 - a) = 1/2\ |\mathbf{x}|^2 := 1/2\ x = t^k$$
$$S_a = 1/2\ (|\mathbf{x}|^2 + a - d(\mathbf{x},\mathbf{x}^r))\ (1-P_0-P_1) + n\ P_1$$

Regrouping the conditions (4.3a) give us relations between the Hamming distance and the failure probabilities P_0 and P_1 which are in fact a trade-off between the possible input faults and the hardware faults:

$$d(\mathbf{x},\mathbf{x}^k) \geq [a - (a+x)(P_0+P_1) + 2nP_1] / (1-(P_0+P_1)) > d(\mathbf{x},\mathbf{x}^r) \qquad (4.3b)$$

Nevertheless, we have to remember that due to the minimum distance classification (Appendix A) the relations

$$d(\mathbf{x},\mathbf{x}^k) > d_H/2 = a > d(\mathbf{x},\mathbf{x}^r)$$

hold.

For the **classification of the class prototypes** itself the condition (4.3b) transforms with $d(\mathbf{x}^k,\mathbf{x}^r) = 2a$ and $d(\mathbf{x}^r,\mathbf{x}^r) = 0$ to the conditions for the maximal tolerable hardware fault probabilities

$$1/4 \geq a/2n \geq P_1 \quad \text{and} \qquad 1 > P_0 + P_1, \qquad P_0 < 1/2 + P_1\ (n/a - 1)$$

The stuck_at_1 connections compensate the stuck_at_0 effects to a certain amount; if we consider only lacking connections ($P_1 = 0$) we can conclude that the device tolerates up to half of the connections beeing left out.

Example: Let n=100, a=10. In the active fault model a proper memory recall of the prototypes is only ensured if P_1 is at most 0.05 and P_0 at most 0.95 .

<u>**Passive fault model**</u>

In the *passive fault model* beside the normally activated connections we have $|\mathbf{x}|^2$ - $\mathbf{xx}^r$ connections which receive x_j=1 but will produce activity only if they are stuck_at_1. With appendix E we have again

$$N_1 = (|x|^2 - xx^r)\, P_1$$

The only difference to N_1 of the active fault model is the term $|x|^2$ instead of n. Therefore, we get nearly the same relation as (4.3b)

$$d(x,x^k) \geq [a - (a+x)(P_0+P_1) + 2xP_1] / (1-(P_0+P_1)) > d(x,x^r) \qquad (4.3c)$$

bearing in mind that

$$d(x,x^k) > d_H/2 = a > d(x,x^r)$$

In the passive fault model for the **recognition of class prototypes** the stuck_at_0 and stuck_at_1 effects are quite symmetric; the condition (4.3c) gives us

$$1/2 \geq P_1 \qquad \text{and} \qquad 1/2 > P_0$$

If we consider the **maximal possible faulty versions** of x^r with $d(x,x^r) < d/2=a$ we get for the passive fault model $P_0=P_1$ and for the active fault model $P_0 = (2n/a - 1)P_1$. In this case no more uncompensated defects can be tolerated; the effects of P_0 and P_1 must compensate each other mutually.

4.4 Discussion

The previous computations in part 4.3 are intended to demonstrate the hardware fault tolerance power, inherent to the non-linear neural network models.
As we can see, the hardware model is very sensitive for active stuck_at_1 faults, i.e. faulty activity, but very robust and fault-tolerant for lacking connections. Thus the fabrication process can be made very easy or the number of connections can be reduced in the design. The whole memory recall process reveals a trade-off in the fault tolerance between the faults in input data and the faults in the hardware.
Nevertheless, we should be still aware of the assumptions under those we have concluded the results above:

♣ the hardware model is very simple. More possible faults will yield a more adequate model. This is especially interesting when you regard a concrete implementation on a chip.

♣ the hardware faults are assumed to be independant. This can be the case when, for instance, stuck_at_0 means the interruption of a connection and stuck_at_1 means the shortcut of an output driver of the connection.
Generally, the two kinds of faults are stochastically dependant due to the different internal failure causes of a connection, leading to the same syndromes stuck_at_0 or stuck_at_1. Without reasonable assumptions for the implementation of a connection this can hardly be quantified.

♣ The threshold condition (3.2i) used here is a special case (normalized prototypes) of the network model. Other models lead to other threshold conditions (e.g. (3.2j)) and therefore to other restrictions for P_0 and P_1.

5.0 Conclusion

In this paper we have investigated the fault tolerance implications for input data and hardware which is caused by the introduction of a non-linearity (threshold) in the linear neural network model.

We computed the necessary and sufficient conditions for memory recall in this pure feed-forward (and free of lateral interaction) non-linear model of associative memory including the optimal thresholds for two different measures of similarity. By the nature of the included threshold mechanism the device gives a proper response even in the presence of erroneous or noisy input data without specially designed fault tolerance support. Thus, the memory recall operation includes a pattern recognition process which can be used for pattern search and pattern completion problems in the field of artificial intelligence. Since the whole operation is done in one clock cycle, the device can be regarded as a *very fast, parallel processor for high-level instructions with inherent fault-tolerance.*

The analysis of the hardware model reveals mainly two effects:

a) The model is quite sensibel for erroneous activity due to active, faulty connections, but very robust and fault-tolerant for the failure or lack of connections. If the implementation of the model chooses only passive connections the resulting design promises to tolerate many faults not only in the normal life cycle but even in the fabrication process.

b) There is a trade-off in the fault tolerance between the faults in input data and the faults in the hardware.

This work was supported by the Stiftung Volkswagenwerk.

References

[COMP] Special issue on Neuron Networks, IEEE Computer, March 1988

[FELD] J.A.Feldman, D.H.Ballard
Computing with connections , University of Rochester, Computer Science Department, TR72, 1980

[FUK] K. Fukushima, A Neural Network Model for selective Attention in Visual Pattern Recognition, Biologigal Cybernetics 55, p.5-15, Springer Verlag 1986

[GOS] K.Goser, C. Foelster, U.Rueckert, Intelligent memories in VLSI. Information Sciences 34, p61-82, 1984

[GROS] S.Grossberg, Contour enhancement, short term memory and constancies in reverberating neural networks, Studies in applied Mathematics 52, pp217-257, 1973

[HILL] D.Hillis, The Connection Machine
MIT Press, Cambridge, Massachusetts, 1985

[HIN1] Hinton, Anderson, Parallel Models of Associative Memory, Lawrence Erlbaum associates, Hillsdale 1981

[HIN2] Hinton, Implementing Semantic Networks in Parallel Hardware, in /HIN1/

[HOPF] J.J.Hopfield, Neural Networks and physical systems with emergent collective computational abilities,
Proc.Natl.Acad.Sci.USA, Vol 79, pp.2554-2558, April 1982

[KOH1] T. Kohonen, Correlation Matrix Memories
IEEE Transactions on Computers C21 1972

[KOH2] Kohonen et alii, A demonstration of pattern processing properties of the optimal associative mapping,
Proc. Int. Conf. Cybernetics and Society, Washington DC 1977

[KOH3] T.Kohonen, Self-Organisation and Associative Memory, Springer Verlag Berlin,New York, Tokyo 1984

[LONG] H.C.Longuet-Higgins,
Holographic model of temporal recall, Nature 217, 1968, p.104

[McCUL] W.S.McCulloch, W.H.Pitts,
A Logical Calculus of the Ideas Imminent in Neural Nets, Bulletin of Mathematical Biophysics Vol 5,1943,pp.115-133

[RUM] Rumelhart, McClelland,
Parallel Distributed Processirtg, MIT Press 1986

[STEIN] K.Steinbuch, Adaptive networks using learning matrices, Kybernetik Vol 2, 1965, pp.148-152

[WIL1] D. Willshaw, Models of distributed associative memory, Unpublished doctoral dissertion, Edinburgh University 1971

[WIL2] D.Willshaw,
Holography, Association and Induction, in /HIN1/

[WIL3] D.Willshaw, O.P.Bunemann, H.C.Longuet-Higgins
Non-holographic associative memory
Nature, 1969, pp.960-962

[WOOD] Wood, Variations on a Theme by Lashley:
Lesion Experiments on the Neural Model.
Psychological Review 85, 1978

Appendix A The class boundary of minimal distance

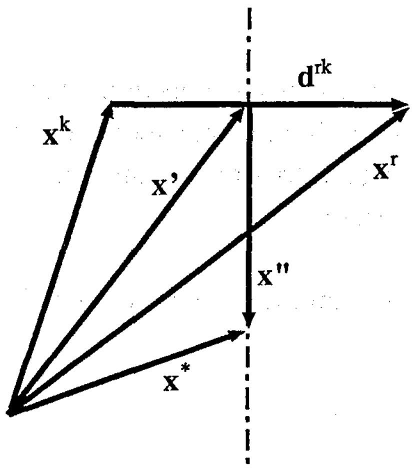
Let {x*} the boundary between two classes r and k which are formed by the classification criterium

$d(x,x^k) > d(x,x^r)$	then x	of class r	*classification by*
$d(x,x^k) < d(x,x^r)$	then x	of class k	*minimal distance*

Theorem:

a) {x*} is a hyperplane which
b) is orthogonal to the distance vector $\mathbf{d}^{rk} := (x^r - x^k)$ and
c) intersects at $\mathbf{d}^{rk}/2$.

Proof:



At the boarder the equation

$$d(x^*,x^r) = d(x^*,x^k)$$

holds.
With $d^2(x^*,x^r) = (x^*-x^r)^2$ we get

$$(x^r)^2 - (x^k)^2 + 2x^*(x^k-x^r) = 0$$

and with (see left figure)
$x' := 1/2\,(x^r-x^k) + x^k$
$x^* := x'' + x'$

we get

$$(x^r)^2 - (x^k)^2 + 2x'(x^k-x^r) + 2x''(x^k-x^r) = 0$$
$$\underbrace{(x^r)^2 + (x^k)^2 - 2x^kx^r - (x^r-x^k)^2}_{0} - 2x''\mathbf{d}^{rk} = 0$$

and so

$$x''\mathbf{d}^{rk} = 0 \quad \text{or} \quad x'' \perp \mathbf{d}^{rk} \quad \text{which prooves a) and b).}$$

For $x^* = x'$ which is the common point of the hyperplane and the distance vector we know that $d(x',x^r) + d(x',x^k) = d(x^r,x^k)$. Because on the boarder the equation $d(x',x^r) = d(x',x^k)$ is also valid, we get $d(x',x^r) = |\mathbf{d}^{rk}|/2$ which is part c) of the theorem.

Appendix B The threshold for maximal correlation

The classification rule (3.2c) is

$$\mathbf{x}\mathbf{x}^r = \max_k \ \mathbf{x}\mathbf{x}^k$$

Let us now regard the seperation of two classes r and k by the classification decision. We know from part 2.1 that the decision boarder is orthogonal to the distance vector $\mathbf{d}^{rk} = |\mathbf{x}^r - \mathbf{x}^k|$. For all $\mathbf{x}$ on the boarder we have

$$\mathbf{x}^*\mathbf{x}^r = \mathbf{x}^*\mathbf{x}^k := \mathbf{x}^*\mathbf{c}^{rk}$$

with $\mathbf{c}^{rk}$ parallel to $\mathbf{x}^*$ (and therefore orthogonal to $\mathbf{d}^{rk}$) as illustrated in figure B.

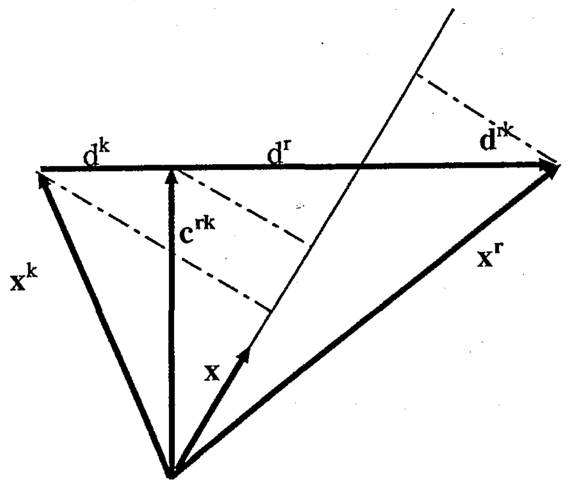


Fig. B boundary of classification with maximal correlation

For a pattern $\mathbf{x}$ we see from figure B that for the projections on $\mathbf{x}$ the relation holds

$$\mathbf{x}\mathbf{x}^r > \mathbf{x}^*\mathbf{c}^{rk} > \mathbf{x}\mathbf{c}^{rk} > \mathbf{x}\mathbf{x}^k \qquad \text{(B.1)}$$

The classification is

$$\begin{array}{ll} \mathbf{x}\mathbf{x}^r > t_{\text{cross}} & \text{then } \mathbf{x} \text{ is of class r} \\ \mathbf{x}\mathbf{x}^r \leq t_{\text{cross}} & \text{then } \mathbf{x} \text{ is not of class r} \end{array} \qquad \text{(B.2)}$$

Certainly, the basic decision criterion for class *r* is $\mathbf{x}\mathbf{x}^r > \mathbf{x}\mathbf{x}^k$. Since our hardware mechanism supports only one comparison without communication between processing elements, the algorithm implied by (3.2c) can not be implemented directly: We can not compare all correlations $\mathbf{x}\mathbf{x}^{k_i}$ of all the other processing elements i with the correlation $\mathbf{x}\mathbf{x}^r$. Instead, we have to choose a threshold t_{cross} which makes a correct decision in one comparison, using only available parameters. For this reason we choose one of the intermediate values of relation (B.1) as threshold, bearing in mind that this narrows the set of patterns belonging to class r. Some patterns of class r are projected on the null vector.

So we choose

$$t^{rk}_{\text{cross}} := \mathbf{x}^*\mathbf{c}^{rk} \quad = |\mathbf{x}||\mathbf{c}^{rk}| \qquad \text{(B.3)}$$

By basic geometric proportions (see fig. B) we have with $|\mathbf{c}^{rk}| =: c$

$$|x^r|^2 = c^2 + (d^r)^2 \qquad (d^{rk})^2 = (d^r + d^k)^2$$
$$|x^k|^2 = c^2 + (d^k)^2$$

and by combining and substitution we get

$$|c^{rk}| = \left(|x^r|^2 - \left((|x^r|^2 - |x^k|^2 + |d^{rk}|^2) / 2d^{rk} \right)^2 \right)^{1/2}$$

or in correlation terms

$$|c^{rk}| = \left(|x^r|^2 - (|x^r|^2 - x^r x^k)^2 / (|x^r|^2 + |x^k|^2 - 2x^r x^k) \right)^{1/2}$$

The threshold must be valid for all classes

$$t^r_{cross} = \max_{k \# r} t^{rk}_{cross} = |x||c^{rk}|$$

Since the threshold part $|c_r|$ can be calculated once before the pattern recall process and stored in the processing element, the resulting threshold can be build up at recall time by calculation of $|x|$ which can be done very easily as shown in figure 3.2c.

For normalized prototypes ($|x^r|^2 = |x^k|^2 =: a$) we get with the maximal cross-correlation K_r

$$t^r_{cross} = \max_{k \# r} |x| \left(1/2(a + x^r x^k) \right)^{1/2} = |x| \left(1/2(a + K^r) \right)^{1/2} \qquad \text{(B.4)}$$

Another threshold may be taken from the distance measure, which has the same decision criterium in the case of normalized prototypes (see C.3).

With $(d^r)^2 = \min_k (x^r - x^k)^2 = \min_k 2(a - x^k x^r) = 2(a - K^r)$

we get by (C.6)

$$t^r_{cross} = 1/4 (2|x|^2 + a + K^r) \qquad \text{(B.5)}$$

Appendix C The threshold for minimal distance classification

The classification of a pattern **x**, to the class of the most resembling prototype x^k is determined by the rule of (3.2d)

$$|x - x^r| = \min_k |x - x^k|$$

With $d(x, x^k) := |x - x^r|$ we have for all classes $k =| \; r$

$$d(x, x^k) > d(x, x^r)$$

$$(x - x^k)^2 = d^2(x, x^k) > d^2(x, x^r) = (x - x^r)^2$$

and so $\quad xx^r > xx^k - 1/2 \, |x^r|^2 + 1/2 \, |x^k|^2$

The classification rule is then

$$\begin{array}{ll} xx^r > t^{rk}_{dist} & \text{then x is of class r} \\ xx^r \leq t^{rk}_{dist} & \text{then x is not of class r} \end{array} \qquad \text{(C.1)}$$

with the threshold

$$t^{rk}_{dist} = \mathbf{x}\mathbf{x}^k - 1/2\,|\mathbf{x}^r|^2 + 1/2\,|\mathbf{x}^k|^2 \qquad \text{(C.2)}$$

For normalized prototypes ($|\mathbf{x}^r|^2 = |\mathbf{x}^k|^2 =: a$) this becomes

$$t^{rk}_{dist} = \mathbf{x}\mathbf{x}^k \qquad \text{(C.3)}$$

which is essentially the cross-correlation criterium.
Let us now calculate a threshold which implements the demand of minimal distance classification. As it is already indicated in appendix B, the threshold which is a decision boarder to all classes will not assign all patterns of class r to the classprototype but some to the null vector.
From appendix A we know that the boundary between two classes r and k is at $d(\mathbf{x}^r,\mathbf{x}^k)/2$.

Thus for $\mathbf{x}$ of class r we have

$$d(\mathbf{x}^r,\mathbf{x}) < d(\mathbf{x}^r,\mathbf{x}^k)/2 < d(\mathbf{x},\mathbf{x}^k)$$

and with condition (2.1b) we get

$$d(\mathbf{x}^r,\mathbf{x}) < \min_k\; d(\mathbf{x}^r,\mathbf{x}^k)/2 < \min_k d(\mathbf{x},\mathbf{x}^k)$$

With $d^r := \min_k d(\mathbf{x}^r,\mathbf{x}^k)$ we have with positive d(.)

$$(\mathbf{x}-\mathbf{x}^r)^2 = d^2(\mathbf{x}^r,\mathbf{x}) < (d^r/2)^2$$
$$\mathbf{x}\mathbf{x}^r > 1/2\,(|\mathbf{x}|^2 + |\mathbf{x}^r|^2 - (d^r/2)^2)$$

The classification rule becomes

$$\begin{array}{lll} \text{If } \mathbf{x}\mathbf{x}^r > t^r_{dist} & \text{then } \mathbf{x} \text{ is of class r} & \\ \text{If } \mathbf{x}\mathbf{x}^r \le t^r_{dist} & \text{then } \mathbf{x} \text{ is not of class r} & \end{array} \qquad \text{(C.4)}$$

with the threshold

$$t^r_{dist} = 1/2\,(|\mathbf{x}|^2 + |\mathbf{x}^r|^2 - (d^r/2)^2) \qquad \text{(C.5)}$$

For normalized protypes this is

$$t^r_{dist} = 1/2\,(|\mathbf{x}|^2 + a - (d^r/2)^2) \qquad \text{(C.6)}$$

In the binary case for the Hamming distance we get

$$t^r_{dist} = 1/2\,(|\mathbf{x}^r|^2 + |\mathbf{x}|^2 - d_H^r/2) \qquad \text{(C.7)}$$

and

$$t^r_{dist} = 1/2\,(|\mathbf{x}|^2 + a - d_H^r/2) \qquad \text{(C.8)}$$

Appendix D Consistent classification by the suprathreshold linear threshold

Theorem: The conditions for suprathreshold linear coupling

$$t^r_i = y_i^r (a-1) \tag{D.1}$$

and $$\mathbf{xx} \leq a-1 \tag{D.2}$$

are necessary and sufficient conditions for a consistent classification of normalized patterns ($|x|^2=a$) composed of natural numbers (x_i of $\{\mathbb{N}\}^n$) by suprathreshold linear transfer functions (3.0a).

Proof:

necessity:

The two conditions (D.1) and (D.2) are equivalent to the two conditions (3.1a) and (3.1b). Since those conditions are the necessary specifications for correct classification corresponding to the transfer function (3.0a) itself, the two conditions (D.1) and (D.2) are also necessary.

sufficiency:

As we have seen in section 3.1 the condition

$$g(\mathbf{x}^r,\mathbf{x}) = \mathbf{x}^r\mathbf{x} - (a-1) > 0 \tag{D.3}$$

is sufficient for a classification of pattern $\mathbf{x}$ to class r.

We now have to show that this classification is consistent, i.e. that the same pattern is not classified also to a class k#r at the same time; all output lines of other classes will rest silent.

From the well-known triangle relation for the distance $d(\mathbf{a},\mathbf{b}) := |\mathbf{a}-\mathbf{b}|$

$$d(\mathbf{x},\mathbf{x}^r) + d(\mathbf{x},\mathbf{x}^k) \geq d(\mathbf{x}^r,\mathbf{x}^k) \qquad |\mathbf{a}| = (\mathbf{aa})^{1/2}\text{ , } |\mathbf{a}|\,|\mathbf{b}| \geq |\mathbf{ab}| = \mathbf{ab}$$

we get

$$d^2(\mathbf{x},\mathbf{x}^r) + 2d(\mathbf{x},\mathbf{x}^r)d(\mathbf{x},\mathbf{x}^k) + d^2(\mathbf{x},\mathbf{x}^k) \geq d^2(\mathbf{x}^r,\mathbf{x}^k)$$

$$(\mathbf{x}-\mathbf{x}^r)^2+2|\mathbf{x}-\mathbf{x}^r||\mathbf{x}-\mathbf{x}^k|+(\mathbf{x}-\mathbf{x}^k)^2 \geq (\mathbf{x}-\mathbf{x}^r)^2+2(\mathbf{x}-\mathbf{x}^r)(\mathbf{x}-\mathbf{x}^k)+(\mathbf{x}-\mathbf{x}^k)^2 \geq d^2(\mathbf{x}^r,\mathbf{x}^k)$$

$$4|\mathbf{x}|^2 - 4\mathbf{xx}^r - 4\mathbf{xx}^k \geq -4\mathbf{x}^r\mathbf{x}^k$$

With (D.3) and (D.2) we get

$$0 \leq |\mathbf{x}|^2 - \mathbf{xx}^r - \mathbf{xx}^k + \mathbf{x}^r\mathbf{x}^k < |\mathbf{x}|^2 - (a-1) - \mathbf{xx}^k + (a-1) = |\mathbf{x}|^2 - \mathbf{xx}^k$$

or

$$\mathbf{xx}^k \leq a-1 \tag{D.4}$$

So we know that for all other classes k the function g(.) results in

$$g(\mathbf{x},\mathbf{x}^k) = \mathbf{xx}^k - (a-1) \leq 0$$

which in turn results in $T(z_i-t_i) = 0$ and therefore in a supression of activity on line i.

Appendix E Evaluation of the failure probability

Suppose we have n connections in one column (cf. fig.2) and a probability of failure of each connection. The probability that among *n* independent elements just *j* are faulty is

$$P(\text{fault})^j \, (1-P(\text{fault}))^{n-j}$$

Since there are $\binom{n}{j}$ such faulty tuples of *j* elements, the probability of *j* faults in *n* elements is

$$P(\text{j faults}) = \binom{n}{j} P(\text{fault})^j \, (1-P(\text{fault}))^{n-j} \qquad (E.1)$$

This is the binomial distribution. The expected number of faulty elements is therefore

$$N = E_{(\text{number of faulty elements})} = \sum_{j=0}^{n} j\, P(\text{j faults}) = \sum_{j=0}^{n} j \binom{n}{j} P(\text{fault})^j \, (1-P(\text{fault}))^{n-j}$$

With P:=P(fault) and Q:=1-P is

$$N = \sum_{j=0}^{n} j \binom{n}{j} P^j \, Q^{n-j} = P \frac{\partial}{\partial P} \sum_{j=0}^{n} \binom{n}{j} P^j \, Q^{n-j} = P\, n\, (P+Q)^{n-1} = nP \qquad (E.2)$$