2 Associative Memory

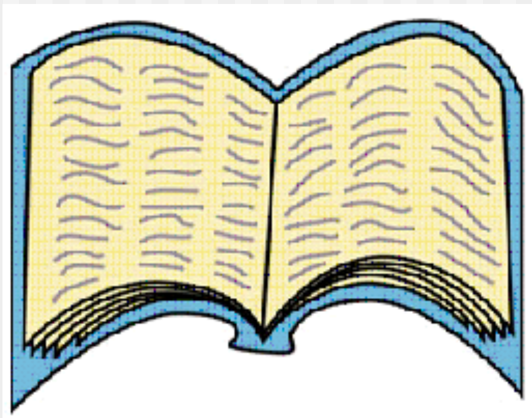
- Content-Addressable Memory
- Associative Memory
  - Lernmatrix
  - Association
  - Heteroassociation
  - Learning
  - Retrieval
  - Reliability of the answer
- Storage Analysis
  - Sparse Coding
- Implementation on a Computer
- Applications
- Hardware

Content-Addressable Memory

- Content-Addressable Memories (CAMs) work differently from traditional memory: stored items are retrieved using their content as a key, rather than using an arbitrary address
- Examples are a phonebook, a search engine or even the router table in an Internet router

Key:

Bryan Davis →



→ *Content:*

(352) 222-5701
305-C NE 6th St

Associative Memory

- Human memory is based on associations with the memories it contains
  - ... Just a snatch of well-known tune is enough to bring the whole thing back to mind
  - ... A forgotten joke is suddenly completely remembered when the next-door neighbor starts to tell it again
- This type of memory has previously been termed content-addressable, which means that one small part of the particular memory is linked - associated -with the rest.

Associative Memory

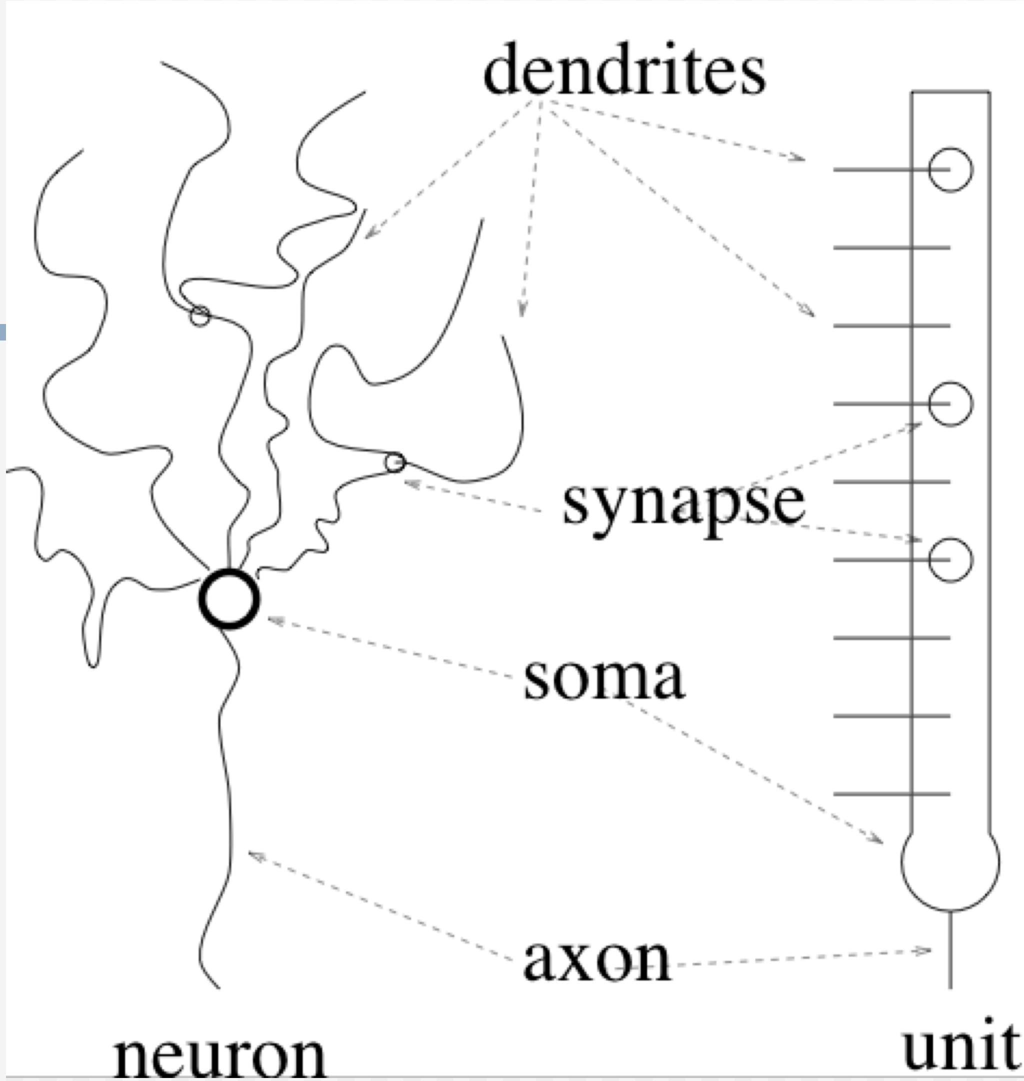
- The ability to correct faults if false information is given
- To complete information if some parts are missing
- To interpolate information, that means if a pattern is not stored the most similar stored pattern is determined

- The cerebral cortex is a huge associative memory
- or rather a large network of associatively connected topographical areas
- Associations between patterns are formed by Hebbian learning

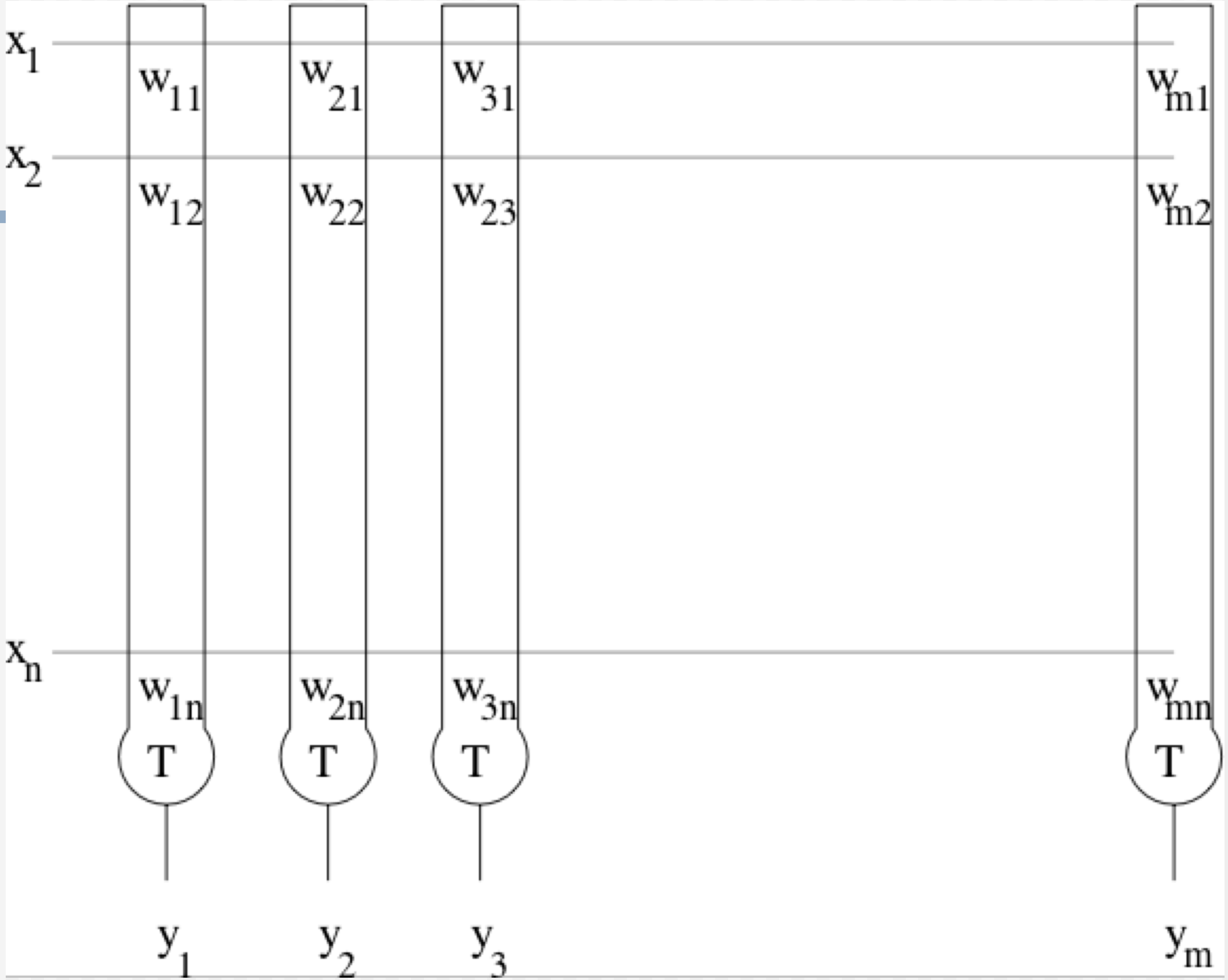
Lernmatrix

- The Lernmatrix, also simply called associative memory was developed by Steinbuch in 1958 as a biologically inspired model from the effort to explain psychological phenomenon of conditioning

- Later this model was studied under the biological and mathematical aspects by G. Palm
- It was shown that Donald Hebb's hypothesis of cell assemblies as a biological model of internal representation of of events and situations in the cerebral cortex corresponds to the formal associative memory model



- The associative memory is composed of a cluster of units which represent a simple model of a real biological neuron



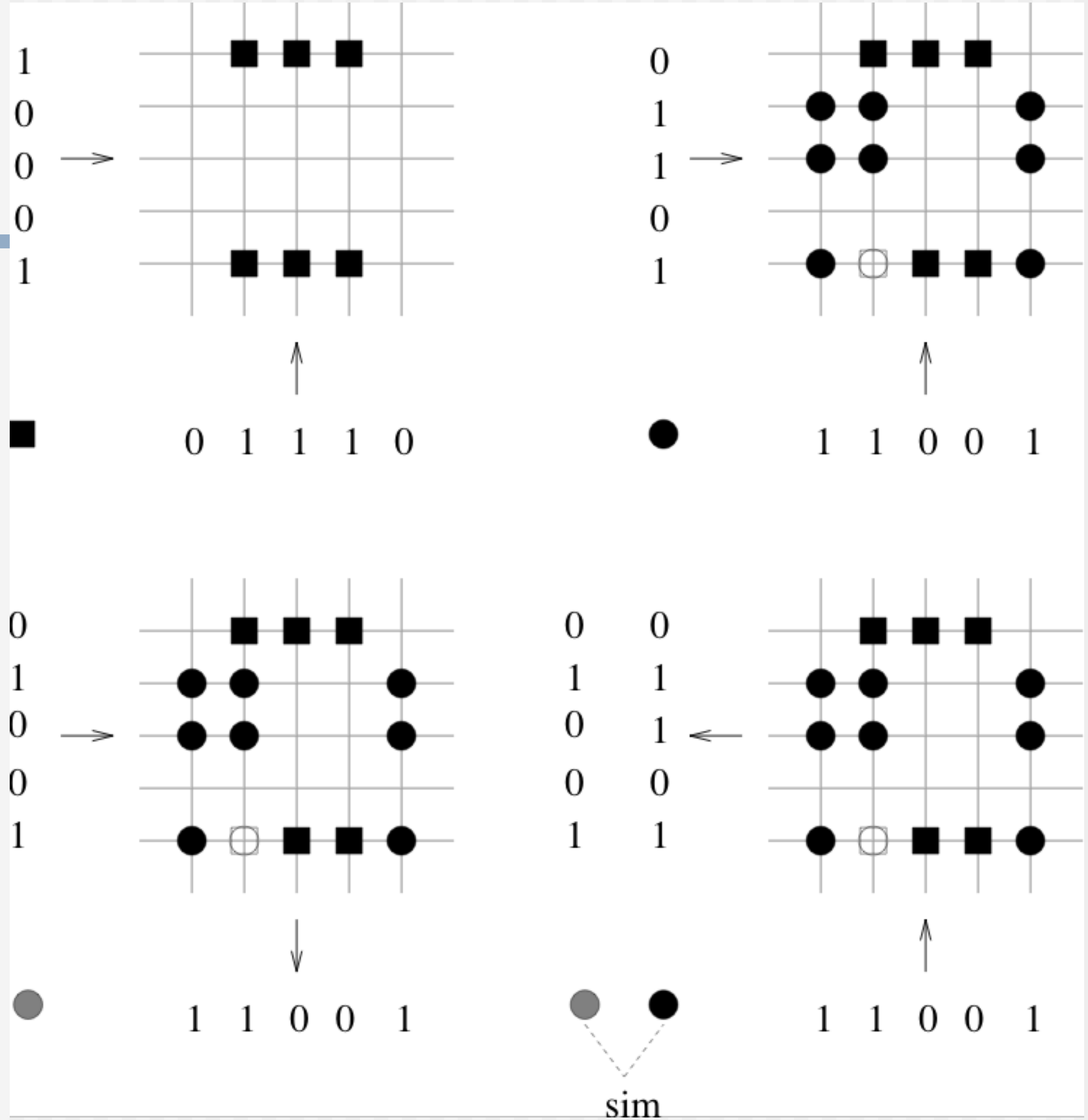
x1
x2
xn
w11
w21
w31
wm1
w12
w22
w23
wm2
w1n
w2n
w3n
wmn
T
T
T
T
y1
y2
y3
ym

- The patterns are represented by binary vectors
- The presence of a feature is indicated by a one component of the vector, its absence through a zero component of the vector

- Always two pairs of these vectors are associated
- This process of the association is called learning

- The first of the two vectors is called the question vector and the second the answer vector
- After the learning the question vector is presented to the associative memory and the answer vector is determined

- This process is called:
  - **association** provided that the answer vector represents the reconstruction of the disturbed question vector
  - **heteroassocation** if both vectors are different


1 0 0 0 1
0 1 1 1 0
0 1 1 0 1
1 1 0 0 1
0 1 0 0 1
1 1 0 0 1
0 1 0 0 1
0 1 1 0 1
1 1 0 0 1
sim

- In the initialization phase of the associative memory no information is stored;

- because the information is represented in the ***w*** weights they are all set to zero

Learning

- In the learning phase, binary vector pairs are associated
- Let ***x*** be the question vector and ***y*** the answer vector, so that the learning rule
- is:

$$w_{ij}^{new} = 1 \quad if\ y_i \cdot x_j = 1$$
$$w_{ij}^{new} = w_{ij}^{old}\ otherwise$$

- This rule is called the binary Hebb rule

- In the *one-step* retrieval phase of the associative memory

- a fault tolerant answering mechanism recalls the appropriate answer vector for a question vector $\boldsymbol{x}$

- To the presented question vector ***x*** the most similar learned $\boldsymbol{x}^l$ question vector regarding the *Hamming distance* is determined
  - Hamming distance indicates how many positions of two binary vectors are different
- The appropriate answer vector ***y*** is identified

Retrieval

$$y_i = \begin{cases} 1 & \sum_{j=1}^{n} w_{ij} x_j \geq T \\ 0 & \text{otherwise.} \end{cases}$$

- *T* is the threshold of the unit
  - In the hard threshold strategy, the threshold *T* is set to the number of “one” components in the question vector
    - If one uses this strategy it is quite possible that no answer vector is determined
  - In soft threshold strategy, the threshold is set to the maximum sum $\sum_{j=1}^{n} \delta(w_{ij} x_j)$

Soft threshold strategy

$$\delta(x) = \begin{cases} 1 \text{ if } x > 0 \\ 0 \text{ if } x = 0 \end{cases}$$

$$T :=_{max\ i} \sum_{j=1}^{n} \delta(w_{ij}x_j)$$

Backward projection

- In this case, ***y*** is the question vector, and the answer vector
- which should be determined is $\boldsymbol{x}^l$

$$x_j^l = \begin{cases} 1 & \sum_{i=1}^{m} w_{ij} y_i \geq T^* \\ 0 & \text{otherwise.} \end{cases}$$

- This means that the synaptic matrix used is a transpose of the matrix ***W*** which is used for the forward projection
    - *T** is the threshold of the unit

Reliability of the answer

- Let ***x*** be the question vector and ***y*** the answer vector that was determined by the associative memory
- First, the vector $\boldsymbol{x}^{l}$ which belongs to the vector ***y*** is determined by a backward projection of the vector ***y***
- The greater the similarity of the vector $\boldsymbol{x}^{l}$ to the vector ***x***, the more reliable the answer vector ***y***

Association learning

$\mathbf{P^{(1)}} = \mathbf{1\;1\;1\;1\;0\;0\;0\;0}$

k i j

$$
\begin{array}{c|ccccccc}
\mathbf{P^{(1)}} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\hline
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\
\mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{1} & 0 & 0 & 0 \\
\mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\mathbf{0} & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

j

i

Association learning

$P^{(1)}$ = 1 1 1 1 0 0 0 0

$P^{(2)}$ = 0 0 1 1 1 1 0 0

k i j

$P^{(1)}$		1	1	1	1	0	0	0	
	$P^{(2)}$	0	0	1	1	1	1	0	j
1	0	1	1	1	1	0	0	0	
1	0	1	1	1	1	0	0	0	
1	1	1	1	1	1	1	1	0	
1	1	1	1	1	1	1	1	0	
0	1	0	0	1	1	1	1	0	
0	1	0	0	1	1	1	1	0	
0	0	0	0	0	0	0	0	0	
	i								

Retrieving

Learned patterns:

$P^{(1)}$ = 1 1 1 1 0 0 0 0

$P^{(2)}$ = 0 0 1 1 1 1 0 0

Address pattern:

P^{X} = 0 1 1 0 0 0 0 0

P^X	A						
0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0

Retrieving

Learned patterns:

$P^{(1)}$ = 1 1 1 1 0 0 0 0

$P^{(2)}$ = 0 0 1 1 1 1 0 0

Address pattern:

P^{X} = 0 1 1 0 0 0 0 0

P^{X}	A						
0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0
x	2	2	2	2	1	1	0

Retrieving

Learned patterns:

$P^{(1)}$ = 1 1 1 1 0 0 0 0

$P^{(2)}$ = 0 0 1 1 1 1 0 0

Address pattern:

P^{X} = 0 1 1 0 0 0 0 0

P^X	A						
0	1	1	1	1	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0
x	2	2	2	2	1	1	0
P^R (T=2)	1	1	1	1	0	0	0

Storage Analysis

- For an estimation of the asymptotic number ***L*** of vector pairs (***x***, ***y***) which can be stored in an associative memory before it begins to make mistakes in retrieval phase.
- It is assumed that both vectors have the same dimension ***n***
- It is also assumed that both vectors are composed of ***M*** *1*s, which are likely to be in any coordinate of the vector

- There are $C(n,M)=\frac{n!}{(n-M)!M!}$ different binary vectors of the dimension ***n*** with ***M*** ones

  - 101
  - 011
  - 110

$$C(3,2)=\frac{3!}{(1)!2!}=\frac{6}{2}=3$$

Information Content

- We can determine for each vector the probability of its presence
- The presence of vector has the same probability
- $p_{C(n,M)}=1/C(n,M)$

$$I = \log_2(u_{C(n,M)}) = \log_2(1/p_{C(n,M)}) = -\log_2(p_{C(n,M)})$$

Entropy in Information since

- Entropy measured in bits

$$I = H(F) = -\sum_i p_i \log_2 p_i$$

- Entropy of L vectors

$$I = H(F) = -L \cdot p_{C(n,M)} \cdot \log_2 p_{C(n,M)}$$

$$I = H(F) = L \cdot \frac{1}{C(n,M)} \cdot \log_2 C(n,M)$$

- Maximize information in correspondence to the size of the associative memory
- Fraction of realized information storage capacity to available information storage capacity

$$Maximize \rightarrow \quad \frac{I}{n^2}$$

$$I := I' = L \cdot \log_2 C(n, M)$$

$$I := L \cdot \log_2 \frac{n!}{(n-M)!M!}$$

$$\textit{Maximize} \rightarrow \quad \frac{I}{n^2}$$

- Depending on the size of *n*, we have to find optimal values for *M* and *L*
  - We have to find two equations

- Probability p of after storing L such binary vectors in the associative memory, that a weight w_{ij} at a certain position (ij) is one

  - Probability $1-p$ after storing L such binary vectors in the associative memory, that a weight w_{ij} at a certain position (ij) is zero
  - For all L pairs of the vectors $x_i y_j=0$
  - For one pair, the probability that a weight is zero corresponds to $(n-M)/n* (n-M)/n$

- For one pair, the probability that a weight is zero corresponds to (n-M)/n* (n-M)/n
- For L pairs, since the probability of an independent sequence of events occurring is the product of events´ individual probabilities:

$$(1-p)=\left[\frac{n^2-M^2}{n^2}\right]^L$$

$$p=1-\left[1-\frac{M^2}{n^2}\right]^L$$

- We try to determine the probability of obtaining an extra 1 during recall of y_k
- We know that the vector x_k has M ones and the probability of a weight being *1* is p
- The probability of getting a spurious/wrong output in p^M, because our input vector x_k has M ones

- Let us demand that the number of spurious/wrong 1s on each y_k vector recall be 1
- The product of *(n-M)*, the number of 0 in y_k and the probability of each 0 being wrongly set to 1 will be set to one

$$(n - M) \cdot p^M = 1$$

$$\text{Maximize} \rightarrow \frac{I}{n^2}$$

$$i) \qquad I = L \cdot \log_2 \frac{n!}{(n-M)!M!}$$

$$ii) \qquad p = 1 - \left[1 - \frac{M^2}{n^2}\right]^L$$

$$iii) \qquad (n-M) \cdot p^M = 1$$

Lets put i) and ii) together to get L

$$(n-M)\left[1-\left[1-\frac{M^2}{n^2}\right]^L\right]^M = 1$$

$$1-\left[1-\frac{M^2}{n^2}\right]^L = (n-M)^{-\frac{1}{M}}$$

$$\left[1-\frac{M^2}{n^2}\right]^L = 1-(n-M)^{-\frac{1}{M}}$$

$$L\cdot\log\left[1-\frac{M^2}{n^2}\right] = \log\left(1-(n-M)^{-\frac{1}{M}}\right)$$

$$L = \frac{\log\left(1-(n-M)^{-\frac{1}{M}}\right)}{\log\left[1-\frac{M^2}{n^2}\right]}$$

- How can we express *C(n,M)*?
  - Logarithmic version of Sterling`s formula is

$$\log(n!) = \frac{n+1}{2}\log(n) - n + \frac{1}{2}\log(2\cdot\pi)$$

$$\log_2(n!) = \frac{n+1}{2}\log_2(n)\cdot\log(2) - n + \frac{1}{2}\log_2(2\cdot\pi)\cdot(\log(2))$$

$$\log_2(n!) = \frac{n+1}{2}\log_2(n)\cdot\log(2) - n + 0.92$$

$$I = L \cdot (\log_2 n! - \log_2 (n-M)!M!))$$

$$I = L \cdot (\log_2 n! - \log_2 (n-M)! + \log_2 (M!))$$

- Using the Sterling`s formula and replacing L we get

$$I = \frac{1}{\log(2)} \left(\frac{\log\left(1 - \left((n-M)^{-\frac{1}{M}}\right)\right)}{\log\left(1 - M^2/n^2\right)} \right) \left(\begin{aligned} &(n+1/2)(\log n - \log(n-M)) + \\ &+ M \cdot \log(n-M) - (M+1/2)\log M - 0.92 \end{aligned} \right)$$

$$I = \frac{1}{\log(2)}\left(\frac{\log\left(1-\left((n-M)^{-\frac{1}{M}}\right)\right)}{\log\left(1-M^2/n^2\right)}\right)\left(\begin{array}{l}(n+1/2)(\log n-\log(n-M))+ \\ +M\cdot\log(n-M)-(M+1/2)\log M-0.92\end{array}\right)$$

$$Maximize \rightarrow \quad \frac{I}{n^2}$$

- Using computer algorithm we find the corresponding values *M* that maximizes *I* depending on *n*, $n=10^2, 10^3, .., 10^{100}$

Storage Analysis

- The optimum value for *M* is approximately

$$M \doteq \log_2(n/4)$$

- *L* vector pairs can be stored in the associative memory

$$L \doteq (\ln 2)(n^2/M^2)$$

- This value is much ***greater*** then *n* if the optimal value for ***M*** is used

Storage Analysis

- *L* is much ***greater*** then *n* if the optimal value for ***M*** is used
- Storage of data ***and*** fault tolerant answering mechanism!
  - Sparse coding: Very small number of 1s is equally distributed over the coordinates of the vectors
  - For example, in the vector of the dimension *n=1000000 M=18*, ones should be used to code a pattern
  - The real storage capacity value is lower when patterns are used which are not sparse

$$L \doteq (\ln 2)(n^2/M^2)$$

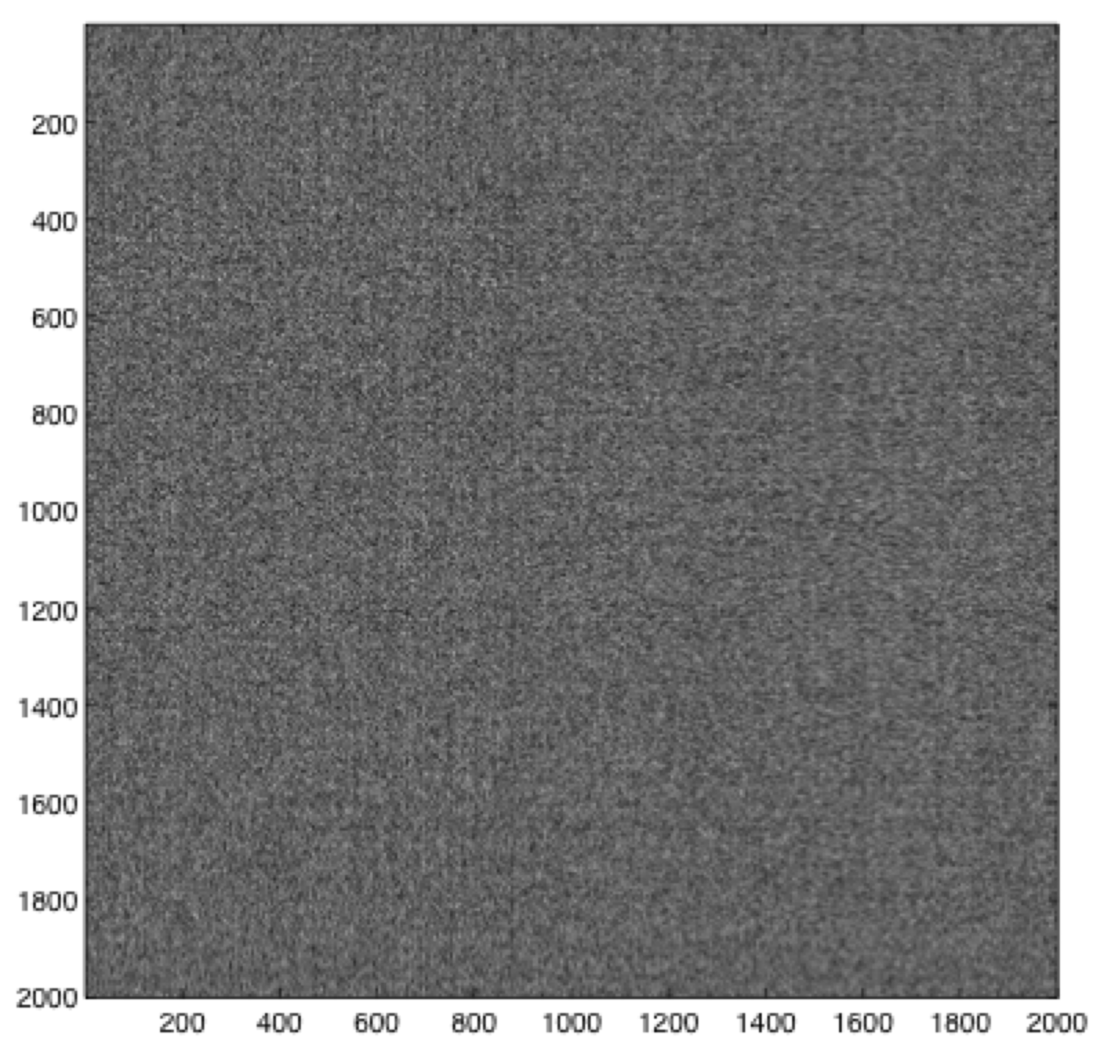
$$L := n$$

$$n = (\ln 2)(n^2/M^2)$$

$$M^2 = (\ln 2)n$$

$$M = \sqrt{(\ln 2)n}$$

- So if $M < 0.8 \cdot \sqrt{n}$ then more then n patterns can be stored



- The weight matrix after learning of 20000 test patterns, in which ten ones were randomly set in a 2000 dimensional vector represents a high loaded matrix with equally distributed weights

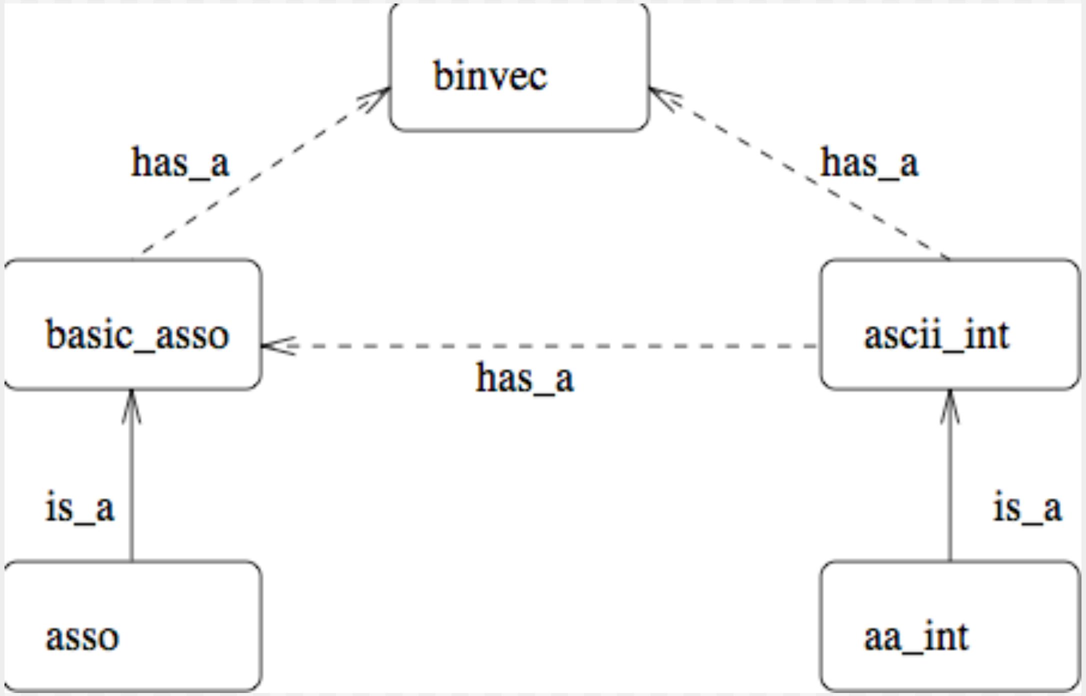
Implementation on a Computer

- On a serial computer a pointer representation can save memory space if the weight matrix is not overloaded
- In the pointer format only the positions of the vector components unequal to zero are represented. This is done, because most synaptic weights are zero. For example, the binary vector [0 1 0 0 1 1 0] is represented as the pointer vector (2 5 6), which represents the positions of "ones"
  - For a matrix each row is represented as a vector

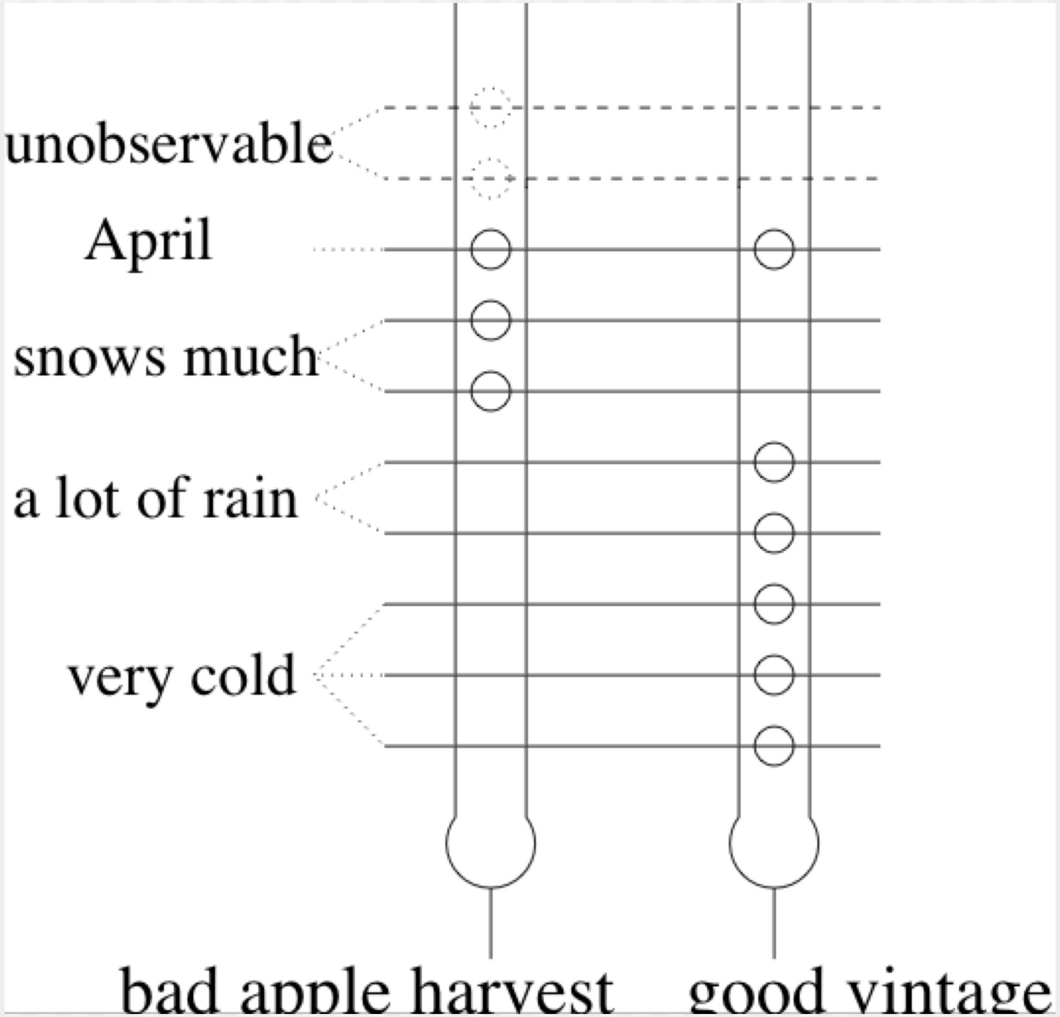
$$\underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{weight\ matrix} \longrightarrow \underbrace{\left(\underbrace{\begin{pmatrix} 1 & \\ 1 & \\ 1 & \\ 2 & 3 \\ 2 & \\ 3 & \end{pmatrix}}_{pointer\ matrix} \ and \ \underbrace{\begin{pmatrix} 2 & \\ 1 & \\ 1 & \\ 1 & 1 \\ 4 & \\ 1 & \end{pmatrix}}_{shadow\ matrix} \right)}_{pointer\ format}$$

Implementation in C++

- http://www.informatik.uni-ulm.de/ni/staff/AWichert.html
- [S2] Wichert A.: Associative Class Library and its Applications, University of Ulm, 1998

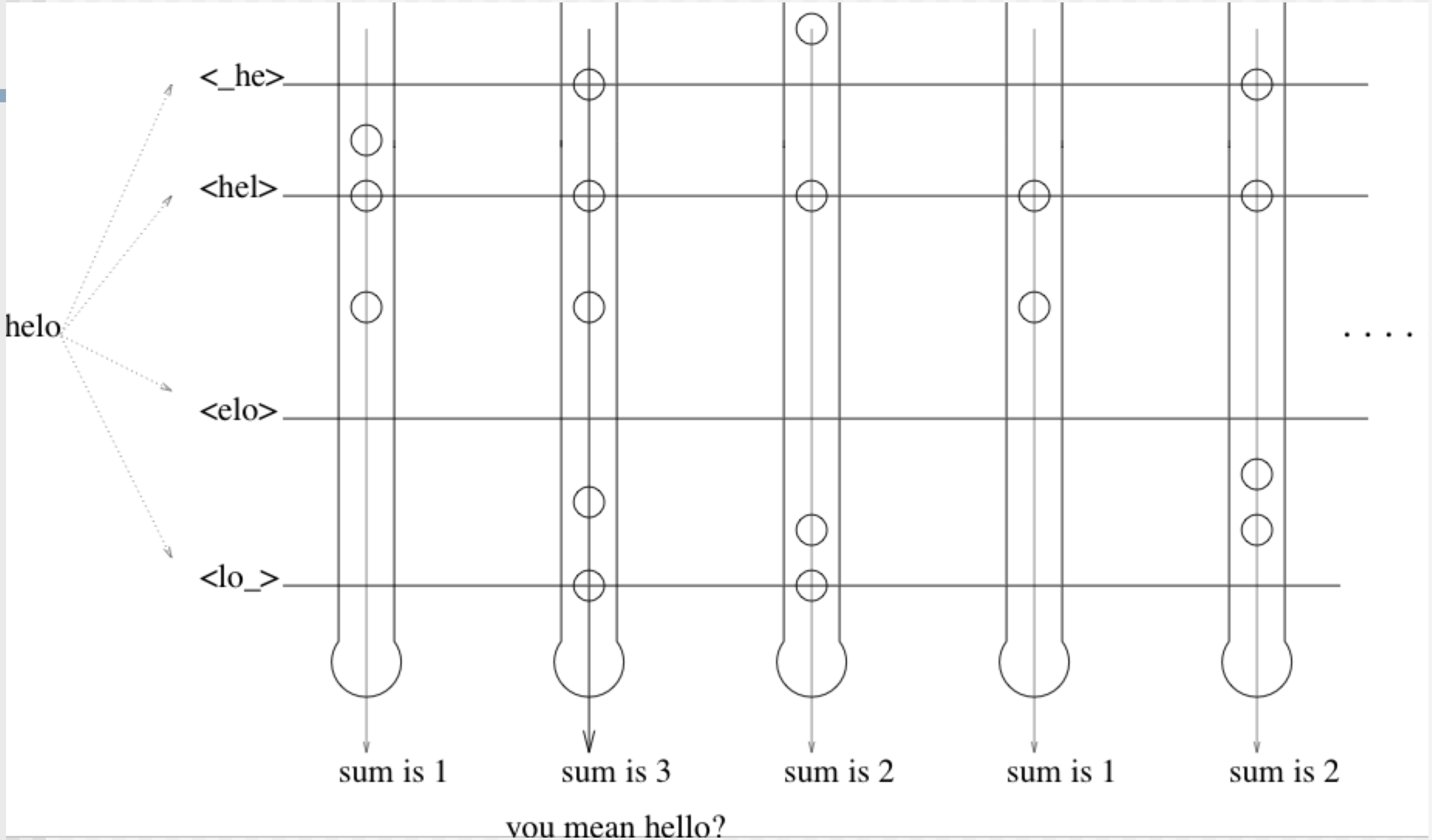


Applications



- Words are represented as sequences of context-sensitive letter units
- Each letter in a word is represented as a triple, which consists of the letter itself, its predecessor, and its successor
  - For example, the word *desert* is encoded by six context-sensitive letters, namely: *_de, des, ese, ser, ert, rt_*
    - The character ` `_" marks the word beginning and ending

- Because the alphabet is composed of *26+1* characters, 27^3 different context-sensitive letters exist
- In the 27^3 dimensional binary vector each position corresponds to a possible context-sensitive letter, and a word is represented by indication of the actually present context-sensitive letters


helo
<_he>
<hel>
<elo>
<lo_>
. . . .
sum is 1
sum is 3
sum is 2
sum is 1
sum is 2
you mean hello?

Coding of answer vector

- *1* of *n* coding
  - Position of the object

- We can use *k* of *n* coding!! (*k* > *n*)
  - 1000, 0100, 00100... (*1* of *n*)
  - 1110, 1101, 11001, ..., 101100,.. (*3* of *n*)

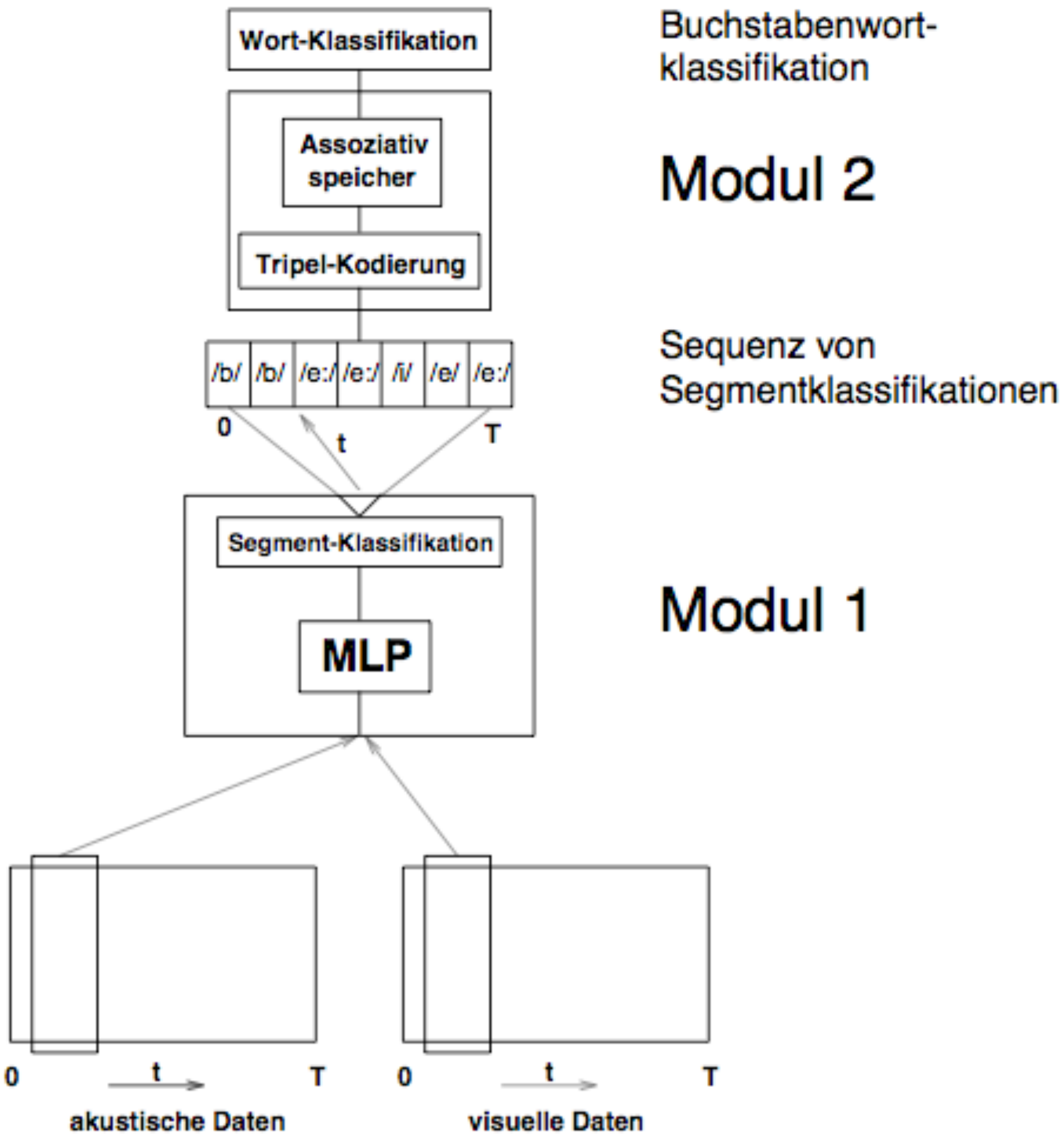
- If sparse coding
  - Then **L > n** !!!!

- A context-sensitive letter does not need to be a triple

- In general, a context-sensitive letter can consist of any number of letters, but only the numbers two, three (**Wickelfeature**) and four letters seem useful

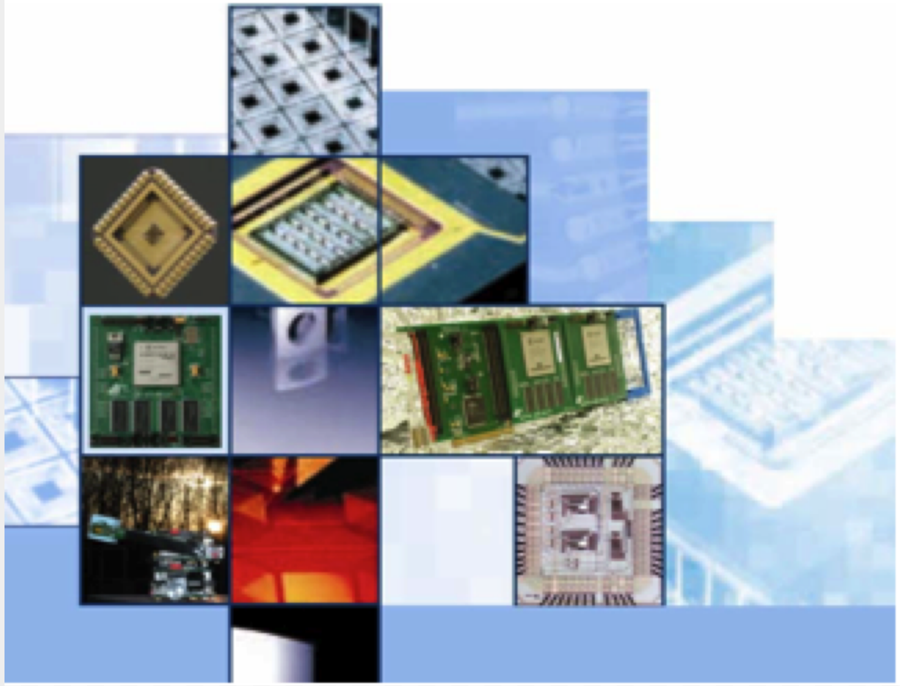
- *Speech system*

- Recognition of visual features and speech features with an artificial neuronal network (Quasiphones)
- Coding of quasiphones by Wickelfeatures
- Recognition of words by associative memory



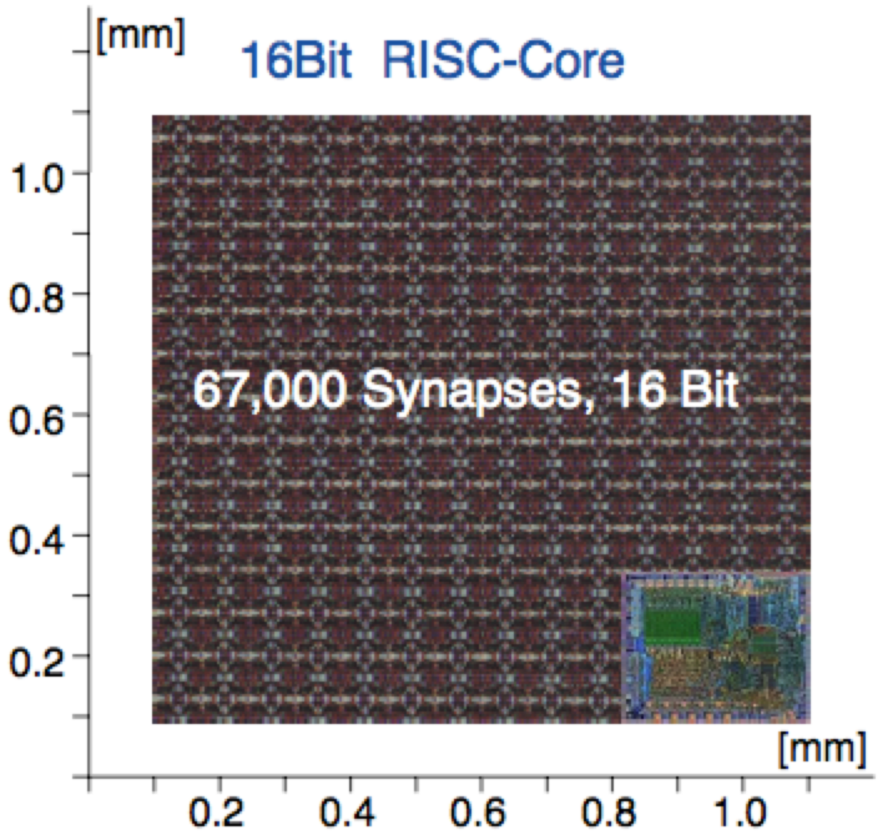
Hardware

Ulrich Rückert

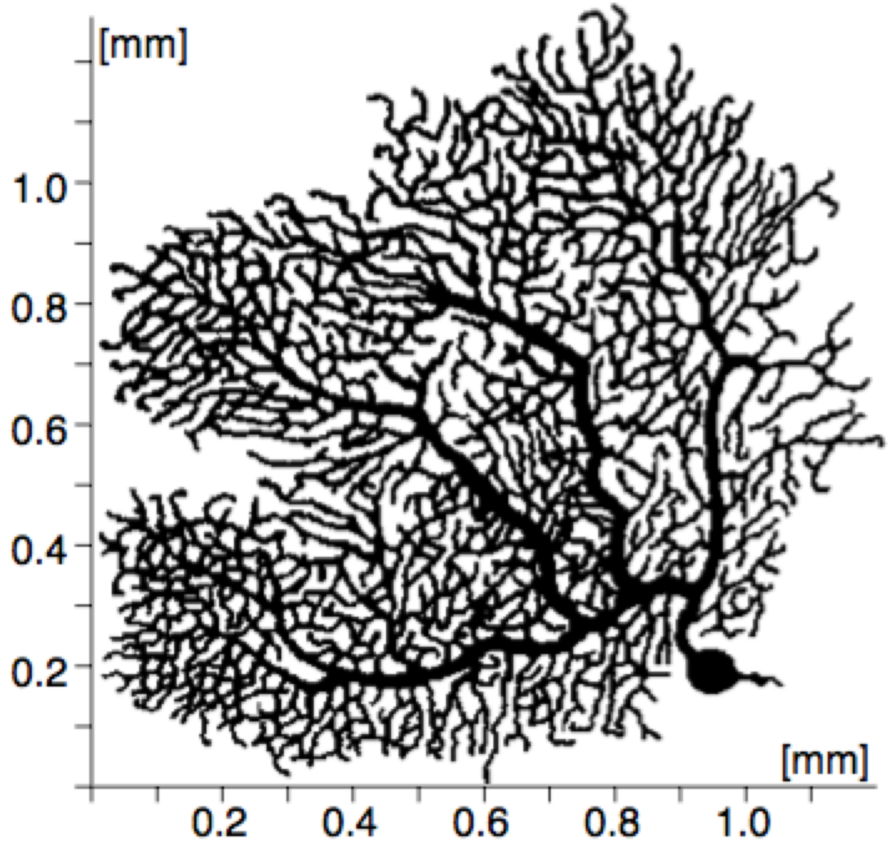


Heinz Nixdorf Institute
System and Circuit Technology
University of Paderborn
Fuerstenallee 11
33102 Paderborn

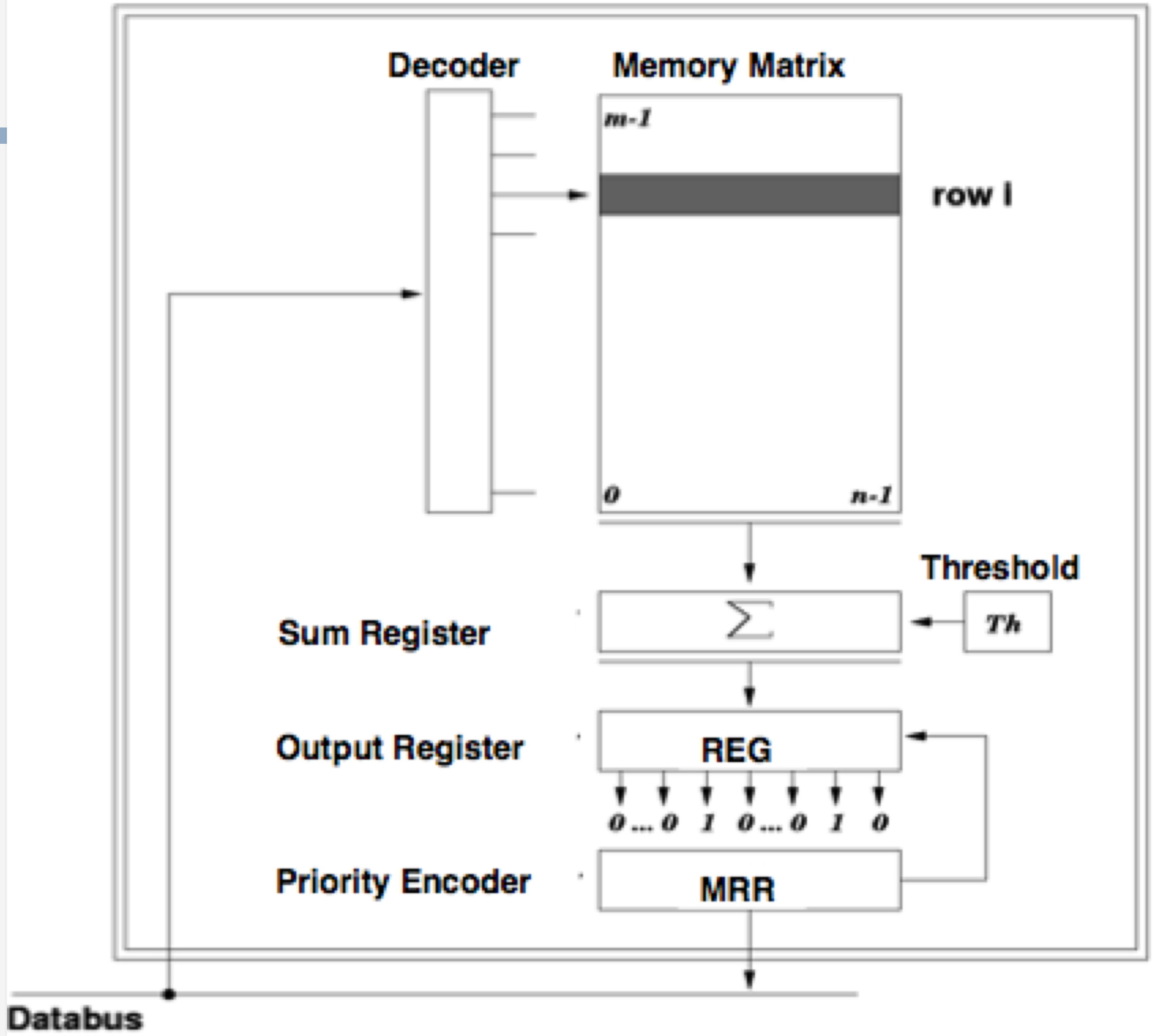
Phone: +49 (0) 52 51/60 6346
Fax: +49 (0) 52 51/60 6351
Email: rueckert@hni.upb.de
http://wwwhni.upb.de/sct/



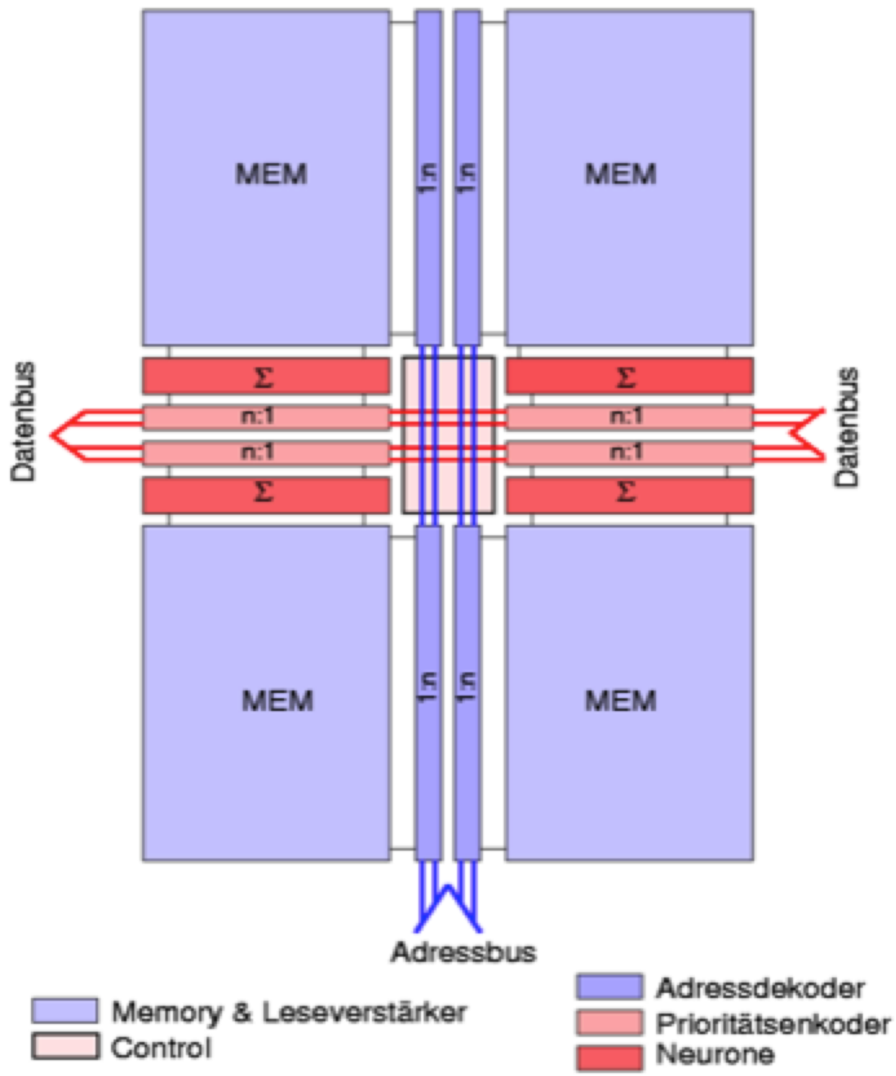
130 nm CMOS

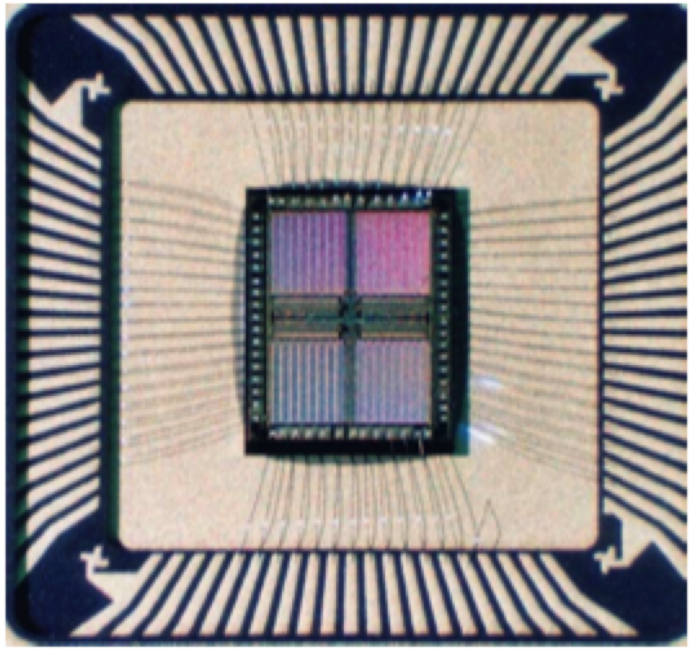


Purkinje cell of cerebellar cortex



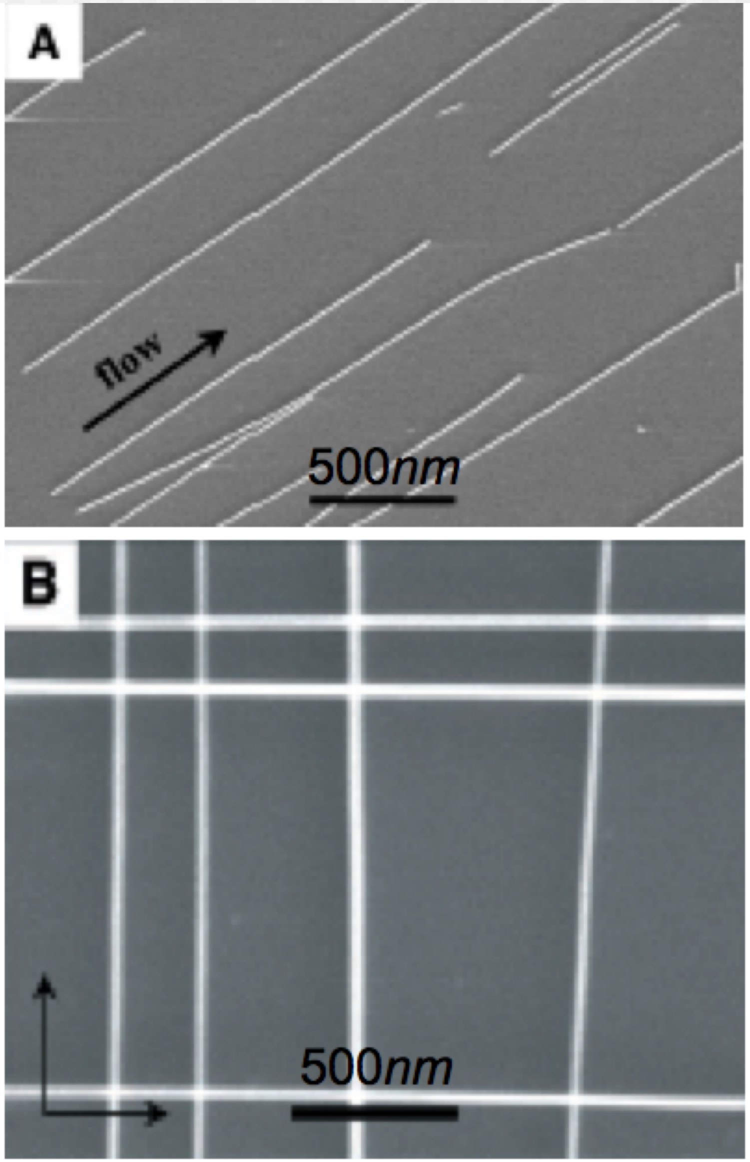
Decoder
Memory Matrix
m-1
row i
0
n-1
Threshold
Sum Register
Th
Output Register
REG
0 ... 0 1 0 ... 0 1 0
Priority Encoder
MRR
Databus





AMS 0.6μm CMOS, 3M
64 Neurone, 8820μm^2 / Neuron
65536 Synapsen
10MHz, 2mW (2V U_{DD})
426MCPS
Chipfläche 18mm^2

These are pictures of nanowires taken with an atomic force microscope. The nanowires are in a suspension, and initially oriented randomly. To form them into a grid, first the suspension flows over a substrate to align the nanowires (the white lines in these pictures) in one direction, as shown in figure A. After the suspension is dried, the solution flows perpendicular to create a grid (figure B). The black scale bar at the bottom of the figures represents 500 nanometers (0.5 micron). (Huang *et al*, *Science,* v. 291, p.630)



UNIVERSITY OF FLORIDA