Brno University of Technology
Faculty of Electrical Engineering and Computer Science
Department of Computer Science and Engineering

Ing. František Grebeníček

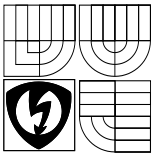
Neural Nets as Associative Memories
Sparse Distributed Memory

Neuronové sítě jako asociativní paměti
Sparse Distributed Memory

PhD Thesis

Brench of study:	Cybernetics and Informatics
Supervisor:	Doc. Ing. František Zbořil, CSc.
Opponents:	prof. Ing. Štefan Hudák, DrSc.
	prof. Ing. Ivo Vondrák, CSc.

Thesis defended on 4th May 2001



Brno 2001

ISBN 80-214-1914-8
ISSN 1213-4198

Contents

1 Introduction

To understand human memory is to understand what concepts are: how they are stored and retrieved, how they overlap with and trigger one another. And it does not take much thought to recognize that understanding the nature of concepts is nothing short of understanding the essence of the mental. Thus, a complete theory of memory would not merely be a theory of one aspect of mind; it would be a complete theory of mind. Pentti Kanerva's theory of associative retrieval and storage represents a significant step in that direction.

Sparse Distributed Memory (SDM) began in 1974 as a paper written for a class on human memory given by Gordon Bower of Standford's psychology department. The main ideas were developed then, and in a couple of years Pentti Kanerva continued working on it until he felt all the bugs were out of it. His monograph named "Sparse Distributed Memory" was published in 1988 [14]. It has lost none of its freshness or originality till today.

Kanerva's SDM can be regarded as a generalized random-access memory wherein the memory addresses and data words come from high-dimensional vector spaces. As in a conventional random-access memory (RAM), there exists an array of storage locations, each identified by a number (the address of the location) with associated data being stored in these locations as binary words. However, unlike conventional RAMs which are usually concerned with addresses and data words only about 32 bits long, SDM is designed to work with address and data vectors with much larger dimensions. Due to the astronomical size of the vector space spanned by the address vectors, it is physically impossible to build a memory containing every possible location of this space. However, it is also unnecessary since only a subset of the locations will ever be used in any application. This provides the primary motivation for Kanerva's model: only a sparse subset of the address space is used for identifying data locations and input addresses are not required to match stored addresses exactly but to only lie within a specified distance (radius) from an address to activate that address.

SDM can be also regarded as a three-layered feed-forward neural network. The main advantages of this net are: (1) learning is very fast, (2) we can understand sense of weights. So, the study of SDM can help us to understand other types of neural nets.

2 The SDM Study

2.1 Associative memories

Associative memory is a memory that can recall data when a reference address is sufficiently closed (not only exact equal as in random-access memories) to the address at which the data were stored. It is very useful if the reference address is corrupted by random noise or outright errors or if this address is only partially specified. *Autoassociative memory* implements a mapping $\Phi(\vec{x}_i) = \vec{x}_i$. *Heteroassociative memory* implements a mapping $\Phi(\vec{x}_i) = \vec{y}_i$.

2.2 SDM as a Neural Net

The SDM may be regarded as an artificial neural net. This net has three layers of neurons — n inputs, L hidden neurons and P output neurons. Input layer neurons copy input vectors only, hidden layer neurons have radial basis functions (RBF) and output layer neurons have linear basis functions (LBF).

Algorithm of weights calculation (for T training pairs (x^t, y^t)) follows. First the weights are initialized:

$$\begin{aligned} w_{jk} = &\begin{cases} 1 & \text{if } Random(1) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} && j = 1 \dots L,\ k = 1 \dots n \\ v_{ij} = &\ 0 && i = 1 \dots P,\ j = 1 \dots L \end{aligned} \tag{1}$$

where $Random(1)$ is a real random number from $\langle 0, 1 \rangle$.

Then, the output weights are computed from the training set:

$$\begin{array}{l} \textbf{for } t = 1 \textbf{ to } T \textbf{ do} \\ \quad \textbf{for } j = 1 \textbf{ to } L \textbf{ do} \\ \quad\quad \textbf{if } \text{Hamming distance}(\vec{x}^t, \vec{w}_j) \leq Radius \textbf{ then} \\ \quad\quad\quad \textbf{for } i = 1 \textbf{ to } P \textbf{ do } v_{ij} = v_{ij} + y_i^t \end{array} \tag{2}$$

where $\vec{w}_j = (w_{j1}, w_{j2}, \dots, w_{jn})$.

The thresholds are given by

$$\Theta_i = \frac{1}{2} \sum_{j=1}^{L} v_{ij} \tag{3}$$

Recall of the net (i.e. reading from the address $\vec{x}$) is done in the similar way:

$$\begin{array}{l}\textbf{for } i = 1 \textbf{ to } P \textbf{ do } u_i = 0\\ \textbf{for } j = 1 \textbf{ to } L \textbf{ do}\\ \quad \textbf{if } \text{Hamming distance}(\vec{x}^t, \vec{w}_j) \leq Radius \textbf{ then}\\ \quad\quad \textbf{for } i = 1 \textbf{ to } P \textbf{ do } u_i = u_i + v_{ij}\\ \textbf{for } i = 1 \textbf{ to } P \textbf{ do}\\ \quad \textbf{if } u_i \geq \Theta_i \textbf{ then } y_i = 1 \textbf{ else } y_i = 0\end{array} \tag{4}$$

SDM Modification It is very difficult to design the SDM for very large input and output vectors ($N \cdot P \geq 1,000,000$) and for technically possible number of hidden neurons. The radius must be very big and the overlappings of hyperspheres are unacceptable large in these cases. On the other hand there is not any active hidden neuron for some input patterns and the net does not produce any response. One way to overcome the described problem may be found in changes of the initial setting of the $\vec{w}_j$ weight vectors.

One of possible modification of the SDM [38] supposes that the number of training pairs T is considerable smaller than that of hidden neurons ($T \ll L$). All hidden neurons are then distributed to T subsets with $\frac{L}{T}$ neurons. The principle of initial setting of the $\vec{w}_j$ weight vectors goes out from the RCE net. These weight vectors of hidden neurons in each subset are generated randomly from unique "parent" input vector with Hamming distances from this parent input vector less than given radius R_g. One of hidden neurons may have $\vec{w}_j$ weight vector equal to parent input vector. The modification is discussed in the section 4.2.1.

2.3 SDM as an Extension of RAM

The idea of distributed storage is that many storage locations participate in a single write or read operation — in marked contrast to conventional computer memories, in which only one location is active at once. Somewhat surprisingly, this gives the memory the appearance of a random-access memory with a very large address space and with data retrieved on the basis of similarity of address. More specifically, if the word ζ is stored at the address a, then reading from a retrieves ζ, and, what is more important, reading from an address x that is sufficiently similar to a retrieves a word ξ that is even more similar to ζ than x is to a (the similarities are comparable because the addresses to the memory and the data are elements of the same metric space, N).

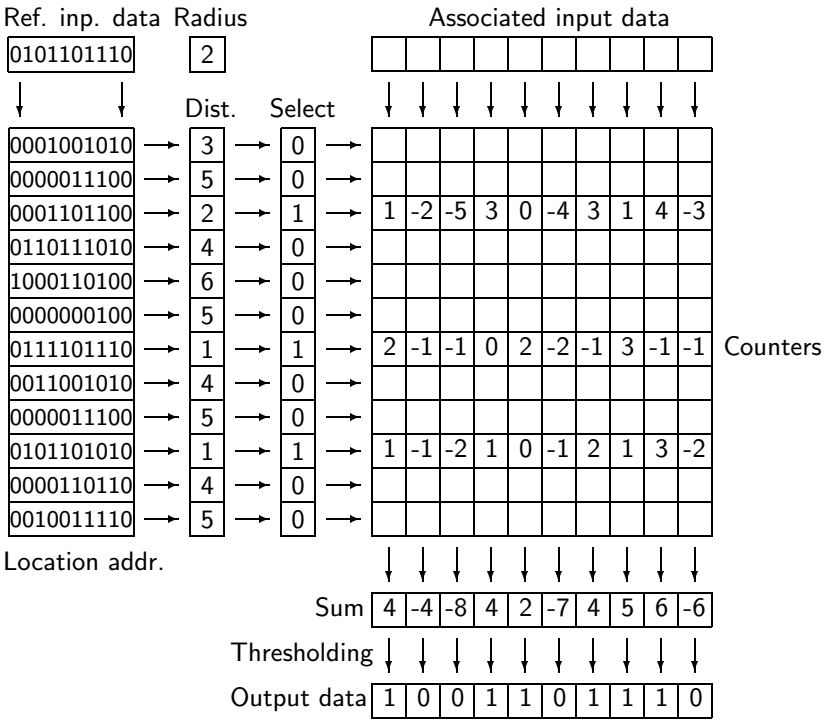


Fig. 1: SDM as an extension of a RAM

The SDM developed by Kanerva may be regarded as an extension of a classical random-access memory (RAM). The main SDM alterations to the RAM are:

- The SDM calculates Hamming distances between the reference address and each location address. For each distance which is less or equal to the given radius the corresponding location is selected.
- The own memory is represented by $L \cdot P$ counters (when L is number of locations and P is the output data length) instead of single-bit storage elements.
- Writing to the memory, instead of overwriting, is done in this way:

  - if the i-bit of the input data is 1, the corresponding counters (counters in the selected locations (rows) and in the i-th colums) are incremented,
  - if the i-bit of the input data is 0, the corresponding counters are decremented.

- Reading (or recall) from the memory is done in a similar way:
  - The contents of the selected locations are summed columnwise.
  - Each sum is thresholded. If the sum is greater than or equal to the threshold value the corresponding output bit is set to 1, in the opposite case it is cleared. Note that the thresholds may be zero, if the training input vectors are closed to orthogonal ones, as is shown in the figure 1.

2.4 Jaeckel-Karlsson's Design

In the *Jaeckel Selected-Coordinate Design* [8] the activation mechanism is based on selecting an activation pattern for each hard location.

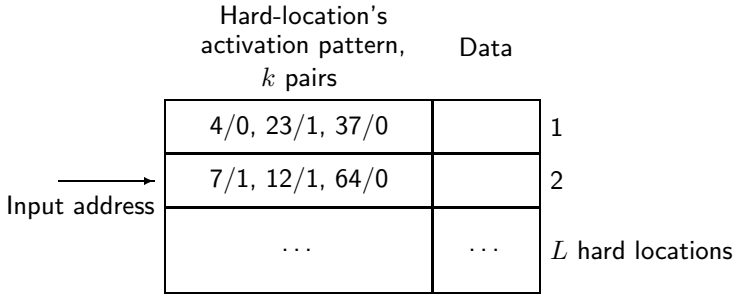


Fig. 2: The Jaeckel Selected-Coordinate Design.

The activation pattern consists of a list of k pairs bit-number/value. The hard location is defined to be "near" the input address if the values (of the activation bits) match exactly. In the example in figure 2, the first hard location is activated by an input address if the bits $4, 23, 37$ in the input address have the values $0, 1, 0$.

If the number of activation bits is k, the probability of activation is $p = 2^{-k}$. If the number of hard locations is L, the mean number of activations per access is $A_{mean} = L \cdot 2^{-k}$.

A table is used in the *Karlsson Selected-Coordinate Design* [15]. The number of entries (A) in the table equals the number of desired activation when accessing the memory with an input address. Each entry in the table specifies a subset of k addres bits. In the example in figure 3 the selected address bits for the first table location are $4, 23, 37$. On access, the specified bits (in the input address) are used as an index to a block of 2^k hard locations. In the example this means that 3 bits in the address are used as an index to one of 8 positions in a block. The hard-location memory consists of A such blocks, i.e., the size of the hard-location memory is therefore $L = A \cdot 2^k$.

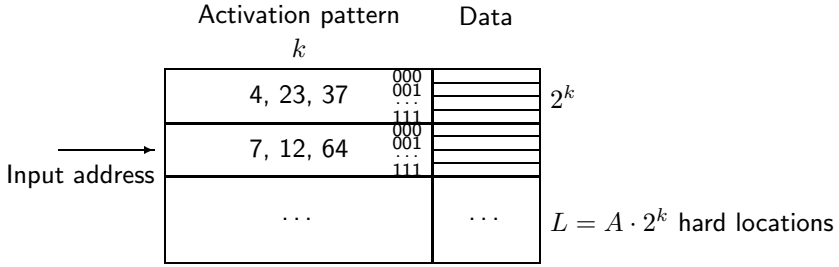


Fig. 3: The Karlsson Design as a restricted Jaeckel Design.

Compared to the earlier designs, the number of access calculations is decreased by a factor 2^k. The number of bits (k) is usually in the range $5 - 20$, making the decrease in access time between 2 and 6 orders of magnitude.

The activation patterns in the table are substantially fewer than in the Jaeckel design (A patterns vs. L patterns). It is not a good idea to generate the activation patterns in the table at random. A bad choice of activation patterns might lead to very unsatisfactory results. In order to remove this problem, a restriction is introduced for choosing bits in the activation patterns: a bit cannot be chosen if there exists any other bit that have been used less times. Note that this optimization is neither necessary nor used in the Jaeckel model.

The Karlsson's design can be viewed as a restricted version of Jaeckel's design (figure 3), with identical functionality (but much slower) since the

Karlsson's design could be implemented by dividing Jaeckel's hard-location memory into A blocks each with 2^k locations, and by using in each block all combination of 0s and 1s in a fixed subset of the address bits.

The two main advantages of the Karlsson's design are: (1) it is much faster and (2) the number of activations is fixed to A. One disadvantage may be that it is less random, e.g., there may be problems with interference between the choice of activation pattern and some input data. One constraint on the basic version of the Karlsson's desing is that the number of hard locations is fixed to $L = A \cdot 2^k$. Of course, this can be remedied by not using all 2^k locations.

3 Dissertation aims

I have defined the aims of my dissertation thesis in three items. The center of work lies in theoretical analysis in the first item and then it drifts to practical analysis in the third item.

3.1 Relation between Data and the Net Efficiency

Adjustment of the net parameters depends on many conditions, for example on the net purpose and the required net merit. Clearly, the radius is a most sensitive net parameter. It can strengthen or weaken a generalization properties of the net. Some other parameters can affect capacity of the net. Analysis of this influence have been shown in [2] and [Gre1]. But, the net behavior also depends on statistical properties of the pattern set, i. e. distances and relations between patterns. Therefore, an optimal parameter adjustment is rather tricky. The aim is to find the net parameters with respect to the pattern set properties.

3.2 SDM Extension

Both the address and data vectors are required to be binary in the standard model of the SDM. An attempt of SDM extension should include integer or real input and output data vectors. There are very important analyses of the extension and the comparison both with standard SDM and with RCE or other classifiers. In some cases, the extension can be found to be equivalent to an existing design.

3.3 Construction of the Hierarchical Networks Consisting of SDMs

A human cerebrum has a hierarchical structure. Designing artificial neural nets, we believe that the best strategy is to learn from the brain itself. We are studying how to synthesize a neural network model which has the same ability as the human brain. As a result of this approach, a pattern-recognition system called the "neocognitron" has been developed (see [4], [5], [27]). The aim is to find a way how to construct similar system using SDM. The system will be probably used in pattern recognition.

4 Research and Results

4.1 Data Analysis

The problem of storing non-random data (with non-uniform distribution) is discussed in this section. We concern to the autoassociative SDM and look for optimal radius value in dependence of statistical data properties.

Distances between pattern addresses are the most interesting data property. If pattern data are random vectors, then (respecting tendency of orthogonality of the space $\{0,1\}^n$) most of them lie about the mean distance $n/2$ from a point. In this case we can get approximately optimal radius value r solving equation:

$$N(r) = 0.01, \quad \text{see Cibulka [2] and Kanerva [14].} \tag{5}$$

Computed value is rounded to integer and SDM working with this radius is maximally efficient. But, pattern data from real applications are not random. It causes less SDM efficiency. So, we look for a way how to find optimal radius for non-random pattern data.

Let $\{t_0 \dots t_{T-1}\}$ is set of T patterns. Let D is matrix of distances between pairs of pattern addresses:

$$D_{ij} = d(t_i, t_j), \quad i, j = 0..T-1, \tag{6}$$

It is clear that $D_{ij} = 0$ for $i = j$. Now we define the mean pattern distance as a mean value of triangular part of matrix D:

$$d_{mean} = \frac{2}{T(T-1)} \sum_{j=1}^{T-1} \sum_{i=0}^{j-1} D_{ij}. \tag{7}$$

We can define the standard pattern deviation in a similar way:

$$d_{dev} = \sqrt{\frac{2}{T(T-1)} \sum_{j=1}^{T-1} \sum_{i=0}^{j-1} (D_{ij} - d_{mean})^2} \tag{8}$$

Hypothesis 1 *The optimal radius is less than the mean pattern distance:*

$$r_{opt} < d_{mean}. \tag{9}$$

In the ideal case, pattern data are random and mean pattern distance approximately equals to $n/2$. Probability $N(n/2) = 1/2$. Clearly, solving equation $N(r) = 0.01$ we get radius less then $n/2$. In the ideal case the hypothesis 1 is performed. We can constitute a stricter hypothesis:

Hypothesis 2 *The optimal radius is less than the mean pattern distance minus the standard pattern deviation:*

$$r_{opt} < d_{mean} - d_{dev}. \tag{10}$$

If pattern vectors are random, the standard pattern deviation approximately equals to $\sqrt{n}/2$. Probability $N(n/2 - \sqrt{n}/2) = F\{-1\} \doteq 0.159$ (F is the normal distribution function). We can solve equation $F\{x\} = 0.01$ for x and find $x \doteq -2.33$. It means that optimal radius for random pattern data is about $d_{mean} - 2.33 d_{dev}$. In the ideal case the hypothesis 2 is performed too. Validity of the hypotheses is demonstrated in [Gre11].

4.2 SDM Modifications

4.2.1 Modifications of Location Addresses Initialization

In real applications, we often do not require the whole address space. Furthermore, for very large input vectors it can be very difficult to generate whole address space with satisfactory number of locations. So, we look for a way to generate only necessary part of the space. One method was proposed by Zbořil in [38] (see section 2.2). In the following, we will analyze the method in details.

Data Driven Distribution (DDD) The proposed modification of the SDM supposes that the number of training pairs T is considerable smaller than the number of hidden neurons ($T \ll L$). All hidden neurons are then distributed to T subsets with $\frac{L}{T}$ neurons. The principle of initial setting of the $\vec{w}_j$ weight vectors goes out from the RCE net. These weight vectors of hidden neurons in each subset are generated randomly from unique "parent" input vector with Hamming distances from this parent input vector less than some given radius r. One of hidden neurons may have $\vec{w}$ weight vector equal to parent input vector [38]. The algorithm of a location generation can be written as:

$$\begin{aligned} &\vec{w}_j = \vec{x}; \\ &\textbf{for } b = 1 \textbf{ to } r \textbf{ do } \{ \\ &\quad k = Random(n); \\ &\quad \text{invert bit } w_{jk} \\ &\} \end{aligned} \tag{11}$$

Figure 4 shows distribution of circle generated by this algorithm for $n = 100$ and radius $r = 40$. The radius is even, then according to the algorithm, only vectors with even Hamming distance can be generated. Mean distance is 27.62 bits, standard deviation 3.45 bits.

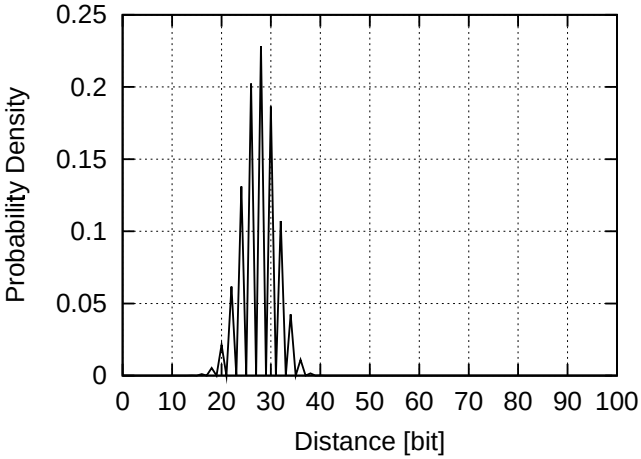


Fig. 4: Distribution of Hamming distance in a circle generated by DDD algorithm 11. $n = 100$, radius $r = 40$.

The whole data driven SDM initialization algorithm can be defined as follows:

$$
\begin{array}{l}
\textbf{for } t = 1 \textbf{ to } T \textbf{ do} \\
\quad \textbf{for } j = 1 + (t-1)L/T \textbf{ to } t \cdot L/T \textbf{ do } \{ \\
\quad\quad \vec{w}_j = \vec{x}_t; \\
\quad\quad \textbf{for } b = 1 \textbf{ to } r \textbf{ do } \{ \\
\quad\quad\quad k = Random(n); \\
\quad\quad\quad \text{invert bit } w_{jk} \\
\quad\quad \} \\
\}
\end{array}
\tag{12}
$$

The algorithm is fast and easy, but has some drawback — generated subspace has distribution very dissimilar to original $\{0,1\}^n$. There are two reasons: first, distribution at figure 4 differs from distribution of circle (see mathematical foundations in [14]), second, circles for different patterns overlap (see figure 7). The overlapping causes this effect: if we generate 100 locations for each pattern and given radius, more then 100 locations will be selected when reading or writing a pattern with the radius. In other words, overlapping increases selection probability in unpredictable way. So, SDM with locations initialized by DDD algorithm does not show the same behaviour as basic Kanerva's SDM.

Binomial distribution We can choose another simple algorithm for generating SDM location address:

$$
\begin{array}{l}
p = r/n; \\
\vec{w}_j = \vec{x}; \\
\textbf{for } k = 1 \textbf{ to } n \textbf{ do} \\
\quad \textbf{if } Random(n) < p \textbf{ then } \text{invert bit } w_{jk}
\end{array}
\tag{13}
$$

This algorithm generates random vectors within the whole space $\{0,1\}^n$ with binomial distribution of Hamming distance (figure 5). Parameters of the distribution are n and $p = \frac{r}{n}$, so the mean value is $np = r$ and variance $npq = r(1 - \frac{r}{n})$. About half of the generated vectors lies within a circle with radius r centered at x. The other vectors are distributed outside the circle. If we use this algorithm for whole SDM initialization, it causes circle overlapping like the DDD algorithm. So, we look for a simple algorithm which allows us to generate a region of $\{0,1\}^n$ with distribution similar to Kanerva's SDM.

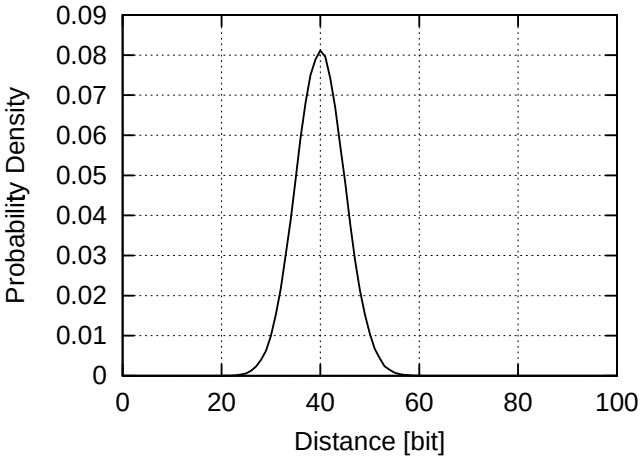


Fig. 5: Binomial distribution of Hamming distance in a circle generated by algorithm 13. $n = 100$, radius $r = 40$.

Selected Uniform Distribution (SUD) The idea of the following algorithm is very simple — we will generate random addresses with uniform distribution and choose only one which lies within given circle:

$$\begin{array}{l}\textbf{repeat}\\ \quad \textbf{for } k = 1 \textbf{ to } n \textbf{ do } w_{jk} = \begin{cases} 1 & \text{if } \mathit{Random}(1) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \\ \textbf{until } d(\vec{w}_j, \vec{x}) \leq r \end{array} \tag{14}$$

Figure 6 shows distribution of circle generated by this algorithm for $n = 100$ and radius $r = 40$. The mean distance is 38.55 bits, standard deviation 1.68. The algorithm can be easily modified to generate all locations according to

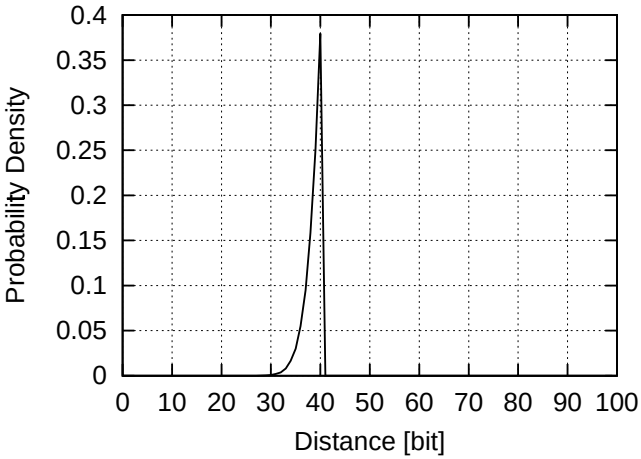


Fig. 6: Distribution of Hamming distance in a circle generated by algorithm 14 "Selected Uniform". $n = 100$, radius $r = 40$.

the whole pattern set:

$$\begin{array}{l}\textbf{for } j = 1 \textbf{ to } L \textbf{ do} \\ \quad \textbf{repeat} \\ \quad\quad \textbf{for } k = 1 \textbf{ to } n \textbf{ do } w_{jk} = \begin{cases} 1 & \text{if } Random(1) > \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \\ \quad\quad selected = \textbf{false}; \\ \quad\quad \textbf{for } t = 1 \textbf{ to } T \textbf{ do} \\ \quad\quad\quad selected = selected \textbf{ or } (d(\vec{w}_j, \vec{x}) \leq r) \\ \quad \textbf{until } selected \end{array} \tag{15}$$

All patterns are passed at once and that is why circle overlapping is restricted (figure 7). The algorithm generates a region of $\{0,1\}^n$ with distribution exactly same as in the Kanerva's SDM. The main disadvantage of an algorithm based on (14) is its very low speed for small radiuses. When radius is reduced to a (relatively) small value, probability of selection decreases rapidly and very large number of loops (attempts) is necessary.

The algorithm (14) can be slightly accelerated by combining with algorithm (13) where probability $p = 2 \cdot R/N(1 - R/N)$ (binomial approximation of circle

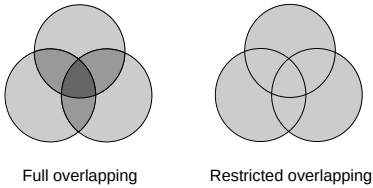


Fig. 7: Circle overlapping

distribution, see [14]):

$$\begin{array}{l} p = 2\frac{r}{n}(1-\frac{r}{n}); \\ \vec{w}_j = \vec{x}; \\ \textbf{repeat} \\ \quad \textbf{for } k = 1 \textbf{ to } n \textbf{ do} \\ \quad\quad \textbf{if } Random(n) < p \textbf{ then} \text{ invert bit } w_{jk} \\ \textbf{until } d(\vec{w}_j, \vec{x}) \leq r \end{array} \tag{16}$$

but the full overlapping problem comes back, because the algorithm can only work with one pattern at once.

Another Address Initialization Some another methods can be used for SDM location address initialization, such as self-organizing techniques [9] or genetic algorithms [1]. This methods are not discussed there.

4.2.2 SDM Efficiency Measuring

This section presents an experiment with artificial pattern set. The experiment should confirm validity of hypotheses introduced in section 4.1.

A pattern set $\{t_0 \dots t_{T-1}\}$ was generated by the algorithm below:

$$\begin{array}{l} \textbf{for } i = 0 \textbf{ to } T-1 \textbf{ do} \\ \quad \textbf{for } k = 1 \textbf{ to } N \textbf{ do } t_{ik} = \begin{cases} 1 & \text{if } Random(1) > q \\ 0 & \text{otherwise} \end{cases} \end{array} \tag{17}$$

where q was so called q-Factor. If $q = \frac{1}{2}$, the patterns are uniformly distributed through the space $\{0,1\}^n$, mean distance $d_{mean} = \frac{n}{2}$, the vectors are orthogonal. When q increases to 1, the mean distance decreases to 0. Figure 8 shows the statistics of pattern sets in dependence on q-Factor: mean pattern

distance, standard distance deviation and the greatest radius solving equation (10) — hypothesis 2. The radius (labeled $H2$) was used for some of tested SDMs.

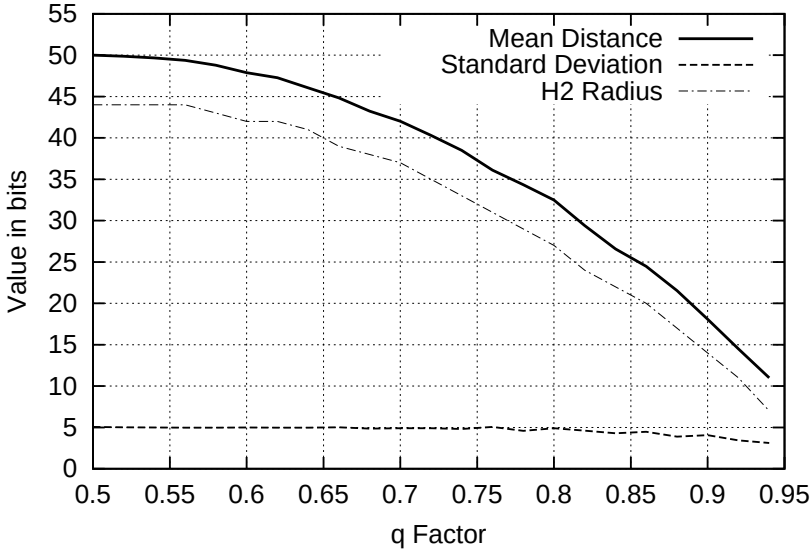


Fig. 8: Statistics of pattern sets

Each of used SDMs had address (and data) width $n = 100$ bits, location count $L = 10,000$. Four SDMs were used in the experiment:

i. Basic Kanerva's SDM with fixed radius $r = 38$ bits. The radius value was calculated solving equation (5) for $n = 100$ bits.

ii. Basic Kanerva's SDM with radius $r = H2$ (depends on used pattern set, see chart at figure 8)

iii. SDM using data driven method of initialization (see section 4.2.1, algorithm 12). Radius $r = H2$.

iv. SDM using selected uniform distribution in initialization (algorithm 15). Radius $r = H2$.

The experiment was composed of a global reading error measuring for all given values of q-Factor. The measuring was done in the following steps:

1. A pattern set was generated for a given q — algorithm 17. The mean pattern distance (7), the standard distance deviation (8) and radius $H2$ (10) were also computed.

2. SDMs were adjusted and re-initialized according to the pattern set.

3. All patterns are written to SDMs.

4. All patterns are read from SDMs. The global error is computed for each SDM as a sum of Hamming distances between written and read vectors.

5. For each SDM, the mean global errors are computed from the global errors. Then, the mean errors should be normalized in dependence of pattern set property. Dividing by the mean pattern distance is used here.

6. Normalized mean errors are written to a log file.

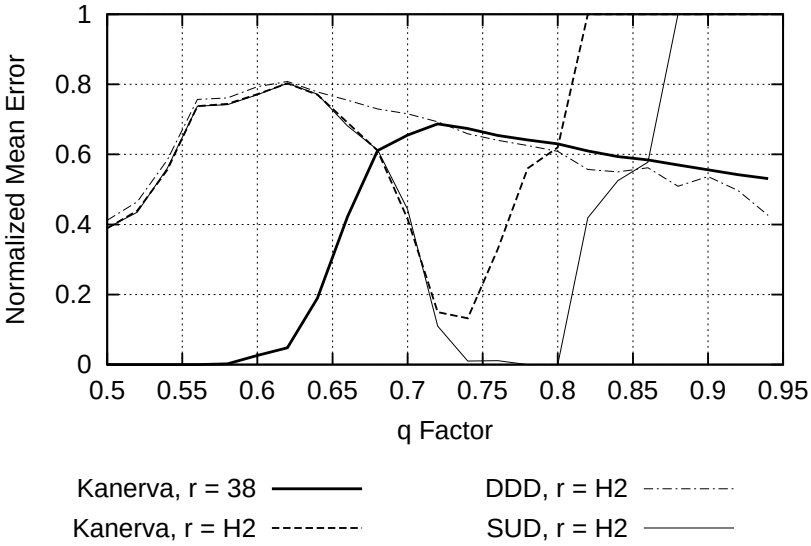


Fig. 9: Results of SDM efficiency measuring

A chart at figure 9 shows results of the experiment. SDM *efficiency* can be considered as inverse normalized mean error, e.g. $\mathit{eff} = 1 - \mathit{err}$, but it is more convenient to use error directly. I have some comments to the chart:

- Basic Kanerva's SDM works well for q from 0.5 to 0.64 (with constant radius $r = 38$ bits).
- All SDMs, except the DDD modification, shows approximately equal efficiency for the radius 38 bits ($q \approx 0.68$). The second and the fourth SDM shows better efficiency than the first (basic Kanerva's) SDM when $q > 0.68$.
- The significant error accumulation of the second SDM for $q > 0.75$ is induced by deficiency of address locations. If radius is less or equal to 24 bits, no location is selected during writing and reading.
- Similarly, error accumulation of the last SDM for $q > 0.8$ is induced by deficiency of address locations. If radius is less or equal to 27 bits, the SUD location generation algorithm (15) is highly time-consuming, so the required number of locations can not be generated.
- Basic Kanerva's SDM shows an interesting effect for $q > 0.72$: normalized error decreases. It can be caused by the generalization ability of SDM — generalized response is better than no response.
- The third SDM (DDD modification) does not get desirable results because of overlapping problem (see figure 7, section 4.2.1, page 18). It should use a less radius than $H2$ for better efficiency. In spite of this, the DDD modification shows the best efficiency for $q > 0.86$.

4.2.3 SDM Working with Vectors of Reals

We need to use SDM with real numbers in the real applications. One possible solution is a data transforming. Another solution is a modification of SDM which can work with vectors of real numbers directly. It is clear that some SDM mechanisms must be slightly changed. I propose following modifications:

- Location addresses are represented by vectors of reals now. Initialization algorithm should distribute them randomly through a hypercube, e.g. $\langle -1, 1 \rangle^n$. This method can be combined with self-organizing techniques. Methods described in section 4.2.1 can be also adapted.

- Hamming distance should be replaced by another metrics, e.g. Euclidean distance.
- Selecting mechanism may be changed. We can add an activation function $A(d) : \langle 0, \infty) \rightarrow \langle 0, 1\rangle$ to the SDM model. Graphs in figure 10 show some of possible activation functions: a) represents the standard Kanerva's activation, b) is the linear fall, c) is a nonlinear fall.

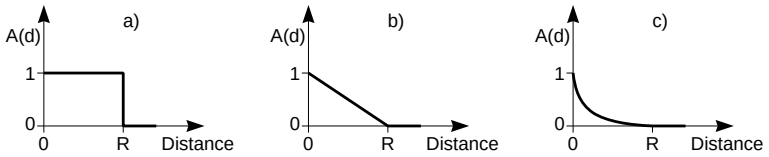


Fig. 10: Some of possible activation functions

- Integer counters should be replaced by real counters. Writing algorithm should distribute data through locations according to activation function.
- Reading algorithm should use the same activation function as the writing algorithm and should compute a weighted average of data from the activated locations.

A possible modification of SDM which implements suggestions above (called Self-Organized SDM) was adapted from Rao [26]. Detailed description and results from prediction applications were published in [Gre9] and [Gre10]. The results showed that a Self-Organized SDM provides an efficient platform for storing and retrieving sequences.

4.3 Construction of Hierarchical Nets

Visual pattern recognition, such as reading characters or distinguishing shapes, can easily be done by human beings, but it is very difficult for a machine. We believe that the best strategy is to learn from the brain itself. The brain seems to have a hierarchical structure with huge parallelism. Simple features are first extracted from a stimulus pattern, and then integrated into more complicated

ones. This section discusses the possibility of constructing hierarchical nets using SDM.

4.3.1 Autoassociative Loop

An SDM may be used as an autoassociative memory. One of simple construction composed of the autoassociative memory is the autoassociative loop. The initial pattern is loaded into the input of the autoassociative SDM through the first input of the multiplexer. The associated pattern is produced on the output of this memory, but it may not be very precise. The output value is therefore loaded to the input again and the loop can continue while some changes in output are detected. Thus, the autoassociative SDM loop produce something like "fall to an attractor". The autoassociative loop was described and tested in an image processsing application [Gre5].

4.3.2 Heteroassociative Loop (Coupled SDMs)

If we insert an heteroassociative SDM into autoassociative loop, we get a very interesting combination of two SDMs — heteroassociative loop (coupled SDMs). It can work in this way: the initial pattern is loaded into the input of the heteroassociative SDM through the first input of the multiplexor. The associated pattern is produced on the output of this memory, but it may not be very precise. The output value of the heteroassociative SDM is therefore loaded to the input of the autoassociative SDM and the precise output pattern is produced on this SDM output now. Obtained output value is passed through the second input of the multiplexor to the input of the heteroassociative SDM as a new input pattern, and so on. Thus, the coupled SDMs produce something as "thinking". Note that order of auto and hetero associative SDMs can be reversed. In this case, the input is corrected by the autoassociative SDM, and then the heteroassociative SDM makes a "step". The heteroassociative loop was described and tested in the "Bonmot" application [Gre2].

4.3.3 Shifted SDM

Classical Kanerva's SDM is tolerant to a random noise in the address. It is a good property, when we use the SDM for the visual pattern recognition. But, we also need a tolerance to image shift and shape warping. Simple SDM does not provide this additional properties. So, we can try to design a hierarchical

net which provides tolerance to the shift and warping. Figure 11 shows a possible suggestion.

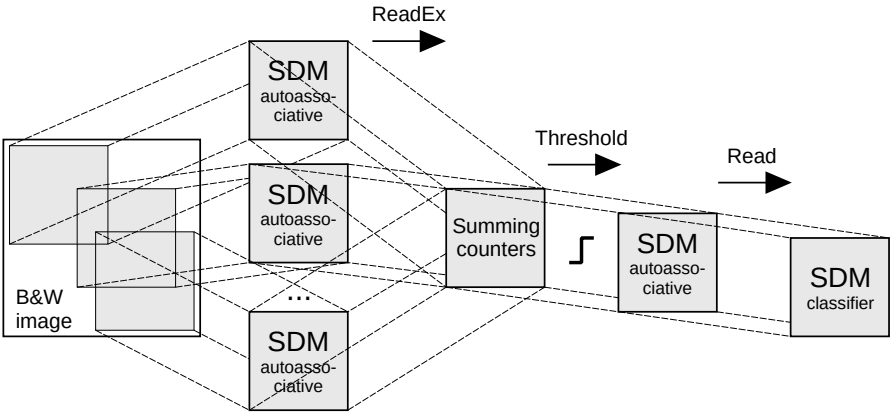


Fig. 11: Hierarchical SDM for visual pattern recognition

A layer of autoassociative SDMs is connected to a black-and-white image. Each of the SDMs scans a square part of the image. Outputs are given by extended reading (without thresholding) and summed in a square array of counters. Then, the array of counters is thresholded to two-level image and passed to the input of single autoassociative SDM which corrects result. Output from the autoassociative SDM is passed to SDM classifier which determines class of the object.

The layer of autoassociative memories consists of identical SDMs. It can be implemented as a sequential shifting of single SDM around the image. It can alse be very easy parallelized (it is a typical single-instruction-multiple-data problem).

The hierarchical net (shifted SDM) was tested in a simple digit recognition application [Gre12].

5 Conclusions

My work was concentrated to the Sparse Distributed Memory as a kind of artificial neural net and associative memory. The SDM captures some basic properties of human long-term memory. Although human memory is much

more complicated than SDM model, it is essential to understand simple models of the right kind before we can home to develop more comprehensive models and to understand the full phenomenon of memory. The SDM model is offered in that spirit.

The first aim of my work was to analyze SDM efficiency in dependence on the pattern data properties. I have studied some statistical properties of pattern data and I have developed two hypotheses (section 4.1) which can be useful for algorithmic SDM adjusting. The hypotheses limits the radius value, the most important SDM parameter. Validity of the hypothesis was shown by efficiency measuring in section 4.2.2 and by a digit recognition application in [Gre11].

The second aim of my work was directed to an extension of SDM. Original SDM model proposed by Kanerva has several weaknesses: it assumes a uniform distribution for the input address vectors, it uses a single fixed threshold (radius) for activating address locations, both the address and data vectors are required to be binary. An extension of original SDM should remove some of weaknesses. In the section 4.2.1 I have analyzed and extend an SDM modification which uses a new method of the location address initialization. The modification was used in many experiments. In the section 4.2.3 I have supposed an SDM modification which can serve vectors of reals in the input and output. Implementation details and results of prediction experiments were published in [Gre9] and [Gre10].

The last aim of my work was the construction of a hierarchical network consisting of SDMs. The brain seems to have a hierarchical structure with huge parallelism. Simple features are first extracted from a stimulus pattern, and then integrated into more complicated ones. Many of complex artificial neural systems are inspired by this strategy. In the section 4.3.3 I have proposed a hierarchical SDM motivated by Fukushima's Neocognitron. The utility of the hierarchical SDM was shown in the digit recognition experiment described in [Gre12].

Bibliography

[1] Anwar, A., Dasgupta, D., Franklin, S.: Using Genetic Algorithms for Sparse Distributed Memory Initialization, CEC 99, URL: `http://www.msci.memphis.edu/~anwara/papers/CEC99_Final.htm` (25 Jul 2000)

[2] Cibulka, J.: Sparse Distributed Memory: An Analysis, Proceedings of the ASIS 1997, Krnov, 1997

[3] Freeman, J.A, Skapura, D.M.: Neural networks, algorithms, applications and programming techniques, Addison-Wesley, 1992

[4] Fukushima K., Wake N.: Handwritten Alphanumeric Character Recognition by the Neocognitron, Neural Networks, Pergamon Press 1991

[5] Fukushima K.: Neocognitron: A Hierachical Neural Network Capable of Visual Pattern Recognition, Neural Networks, Pergamon Press 1988

[6] Hassoun, M. H.: *Fundamentals of Artificial Neural Networks*, The MIT Press, 1995

[7] Haykin, S.: *Neural Networks: A Comprehensive Foundation*, Macmillan/IEEE Press, 1994.

[8] Jaeckel L. A.: An Alternative Design for a Sparse Distributed Memory, Technical report, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989

[9] Kačer, P.: Sparse Distributed Memory — Analysis and Applications [MSc. thesis], Brno UT, 2000

[10] Kanerva, P.: Encoding Structure in Boolean Space, In L. Niklasson, M. Boden, and T. Ziemke (eds.): ICANN 98 Proceedings, pp. 387 – 392. Springer, Berlin, 1998

[11] Kanerva, P.: Fully distributed representation, 1997 Real World Computing Symposium (RWC'97) Proceedings, pp. 358 – 365

[12] Kanerva, P.: Large Patterns Make Great Symbols: An Example of Learning from Example, To appear in NIPS98 workshop proceedings

[13] Kanerva, P.: Pattern completion with distributed representation, The WCCI'98/ IJCNN'98 conference, Anchorage, U.S.A., 1998

[14] Kanerva, P.: *Sparse Distributed Memory*, The MIT Press 1988, Cambridge, ISBN 0-262-11132-2

[15] Karlsson, R.: A Fast Activation Mechanism for the Kanerva SDM Memory, SICS Technical Report R95-10, 1995, ISRN SICS-R–95/10-SE, ISSN 0283-3638.

[16] Karlsson, R.: Compensating for Bias in the SDM Fast Activation Mechanism, SICS Technical Report R96-04, 1996, ISRN SICS-R–96/04-SE, ISSN 0283-3638

[17] Kristoferson, J.: Best Probability of Activation and Performance Comparisons for Several Designs of Sparse Distributed Memory, SICS Technical Report R95-09, 1995, ISRN SICS-R–95/09-SE, ISSN 0283-3638

[18] Kristoferson, J.: Some Comments on the Information Stored in Sparse Distributed Memory, SICS Technical Report R95-11, 1995, ISRN SICS-R–95/11-SE, ISSN 0283-3638

[19] Kristoferson, J.: Some Results on Activation and Scaling of Sparse Distributed Memory, SICS Research Report R97-04, 1997, ISRN SICS-R–97/04-SE, ISSN 0283-3638

[20] Kristoferson, J.: Some Results on Activation and Scaling of Sparse Distributed Memory, In A. de Padua Braga and T. B. Ludermir (eds.): SBRN'98 Proceedings, pp. 157 – 160. IEEE Computer Society, U.S.A, 1998

[21] Looney, C. G.: *Pattern recognition using neural networks: theory and algorithms for engineers and scientists*, Oxford University Press, Inc., New York, 1997

[22] Novák, M., a kolektiv: *Umělé neuronové sítě, teorie a aplikace*, C. H. Beck, Praha, 1988

[23] Novák M., Faber, J., Kufudaki, O.: *Neuronové sítě a informační systémy živých organismů*, Grada, Praha, 1992

[24] Poggio, T., Girosi, F.: Networks for approximation and learning, Proc. IEEE, 78: 1481-1497, 1990

[25] Rogers, D.: Kanerva's sparse distributed memory: An associative memory algorithm well-suited to the Connection Machine. In Proc.

Conference on Scientific Application of the connection machine, Moffet Field, CA, 1988, Vol.1, 282-298.

[26] Rao, R. P. N., Fuentes, O.: Learning Navigational Behaviours using a Predictive Sparse Distributed Memory, Proceedings of From Animals to Animats: The Fourth International Conference on Simulation of Adaptive Behavior, MIT Press, 1996.

[27] Semela, R.: Rozpoznávání ručně psaných znaků [Ročníkový projekt], VUT FEI UIVT 1995

[28] Sjödin, G., Kanerva, P., Björn, L., Kristoferson, J.: Holistic Higher-level Structure-forming Algorithms, In 1998 Real World Computing Symposium (RWC'98) Proceedings, pp. 299-304

[29] Sjödin, G., Karlsson, R., Kristoferson, J.: Algorithms for efficient SDM, 1997 Real World Computing Symposium (RWC'97) Proceedings, pp. 215-222

[30] Sjödin, G.: Convergence and new operations in SDM, SICS Technical Report R95-13, 1995, ISRN SICS-R–95/13-SE, ISSN 0283-3638

[31] Sjödin, G.: Convergence in Sparse Distributed Memory, In A. de Padua Braga and T. B. Ludermir (eds.): SBRN'98 Proceedings, pp. 165 – 168. IEEE Computer Society, U.S.A, 1998

[32] Sjödin, G.: Getting more information out of SDM, In C. von der Malsburg, W. von Seelen, J.C. Vorbruggen, and B. Sendhoff (eds.), Artificial Neural Networks – ICANN 96 Proceedings, pp. 477 – 482. Springer, Berlin, 1996

[33] Sjödin, G.: Improving the Capacity of SDM, SICS Technical Report R95-12, 1995, ISRN SICS-R–95/12-SE, ISSN 0283-3638.

[34] Sjödin, G.: The Sparchunk Code: A Method to Build Higher-level Structures in Sparsely Encoded SDM, The WCCI'98/IJCNN'98 conference, Anchorage, U.S.A., 1998

[35] Skapura, D. M.: Building Neural Networks, ACM Press, New York, 1996

[36] Šíma, J., Neruda, R.: *Teoretické otázky neuronových sítí*, MATFYZ-PRESS, 1996

[37] Zbořil, F.: An Application of the Sparse Distributed Memory, Proceedings of the ASIS 1997, Krnov, 1997

[38] Zbořil, F.: Sparse Distributed Memory and Restricted Coulomb Energy Classifier, In: Proceedings of the MOSIS'98, MARQ, Ostrava, Sv. Hostýn – Bystřice pod Hostýnem, 1998, s. 171 – 176, ISBN 80-85988-23-2

[39] Zbořil, F.: Analysis of Neural Networks Applications in Heterogeneous Models, In: Proceedings of the ASIS'98, MARQ Ostrava, Krnov, 1998, p. 21 – 26, ISBN 80-85988-26-7

[40] Zbořil, F.: Neural Associative Memories, In: Proceedings of the MOSIS'99, MARQ Ostrava, Rožnov pod Radhoštěm, 1999, p. 131–136, ISBN 80-85988-33-X

Author Bibliography

[Gre1] Zbořil, F., Cibulka, J., Grebeníček, F.: Neuronové asociativní paměti, final technical report FR VŠ 627/97, Brno, 1997

[Gre2] Grebeníček, F.: Data coding for SDM, In: Mosis'98 Proceedings, vol. 1, MARQ, Ostrava, Hostýn, Czech Republic, 1998, p. 149–154, ISBN 80-85988-23-2

[Gre3] Grebeníček, F., Dao, A., Ondráček, T., Zbořil, F.: Aplikace neuronových asociativních pamětí, final technical report FR VŠ 191/98, 24 stran, Brno, 1998

[Gre4] Grebeníček, F.: Computer Aided Cytological Examination, In: ECI'98 Proceedings, FEEI TU Košice, Košice - Herľany, Slovakia, 1998, p. 222–227, ISBN 80-88786-94-0

[Gre5] Grebeníček, F.: Neural Net Applications in Image Processing, In: Asis'98 Proceedings, vol. 2, MARQ, Ostrava, Krnov, Czech Republic, 1998, p. 27–32, ISBN 80-85988-27-5

[Gre6] Grebeníček, F.: Využití neuronových asociativních pamětí při zpracování obrazu, Sborník prací studentů a doktorandů, FEI VUT Brno, Brno, 1998, p. 43–44, ISBN 80-214-1141-4

[Gre7] Grebeníček, F.: Educational Models of Neural Nets, In: IWCIT'99, vol. 1, Technical University of Ostrava, Ostrava, Czech Republic, 1999, p. 92–97, ISBN 80-7078-679-5

[Gre8] Grebeníček, F.: Educational Models of Selected Neural Net Types, In: MOSIS'99 Proceedings, vol. 2, MARQ Ostrava, Rožnov pod Radhoštěm, Czech Republic, 1999, p. 77–83, ISBN 80-85988-33-X

[Gre9] Grebeníček, F.: Self-Organizing Sparse Distributed Memory as a Predictive Memory, In: Nostradamus '99, TU Brno, Faculty of Technology Zlin, Zlín, 1999, p. 17–22, ISBN 80-214-1424-3

[Gre10] Grebeníček, F.: Self-Organized Sparse Distributed Memory — an Application, In: ASIS'99, MARQ, Krnov, 1999, p. 39–44, ISBN 80-85988-41-0

[Gre11] Grebeníček, F.: Sparse Distributed Memory — Pattern Data Analysis, In: MOSIS 2000 Proceedings, vol. 1, MARQ Ostrava, Rožnov pod Radhoštěm, Czech Republic, 2000, p. 165–170, ISBN 80-85988-44-5

[Gre12] Grebeníček, F.: Constructing Hierarchical Neural Nets Using Sparse Distributed Memory, In: ASIS 2000 Proceedings, MARQ Ostrava, Hostýn, Czech Republic, 2000, ISBN 80-85988-51-8

[Gre13] Grebeníček, F.: Sparse Distributed Memory — Modifications of Initialiation, to appear in Neural Network World, Academy of Sciences of the Czech Republic, Praha, 2001

Author's Curriculum Vitae

- Name: **František Grebeníček**, MSc.
- Born: 1973, Uherské Hradiště
- Nationality: Czech

Education

- 1997 – 2000 PhD studies, Dept. of Computer Science and Engineering, FEECS, Brno University of Technology
- 1997 MSc. in computer science and engineering, FEECS, Brno University of Technology
- 1992 Secondary Technical School, Uherské Hradiště

Professional Career

- March 1999: two weeks scholarship at Lappeenranta University of Technology, Finland
- In 1995 – 1997 he was concentrated on biological pattern recognition. He participated on the project of computer aided cytogenetic examination. This project was implemented in Department of Genetics, Veterinary Research Institute in Brno.
- Teaching: Artificial Intelligence, Neural Networks

Other

- Languages: English (intermediate)
- Technical interests: artificial intelligence, pattern recognition, neural nets, Java programming
- Personal interests: origami

Abstrakt

Práce pojednává o druhu neuronové asociativní paměti nazývané *Sparse Distributed Memory* (SDM). Teorii SDM rozvinul Pentti Kanerva ve své monografii [14] vydané v roce 1988. Kanervovu SDM můžeme považovat za zobecněnou paměť RAM. Stejně jako u konvenční RAM, existuje u SDM pole tzv. nosičů, z nichž každý je identifikován svou adresou a uchovává asociovaná data. Zatímco konvenční RAM běžně pracuje s vektory o šířce maximálně 32 bitů, SDM umožňuje pracovat vektory širokými stovky až tisíce bitů. Základní myšlenkou Kanervova modelu je využít jenom podmnožinu adresového prostoru. Při čtení z paměti nemusí vstupní adresa přesně odpovídat adrese, na které jsou uložena data – k aktivaci nosiče stačí, když vstupní adresa leží v blízkém okolí (definovaném globálním poloměrem) adresy nosiče. Tento mechanismus odpovídá neuronovým sítím s radiální bázovou funkcí – SDM lze popsat jako třívrstvou dopřednou neuronovou síť. Výhodou SDM je rychlé učení, navíc dovedeme interpretovat váhy mezi neurony. Studium SDM tedy může přispět i k pochopení jiných typů sítí.

Práce měla tři cíle:

1. *Analyzovat chování SDM v závislosti na statistických vlastnostech zapisovaných dat.* SDM pracuje nejlépe s daty náhodnými - s adresami rovnoměrně rozloženými po celém adresovém prostoru. Za těchto předpokladů lze nalézt vztah pro optimální velikost poloměru, který závisí pouze na šířce adresy. Pokud ale data nejsou náhodná, je třeba hledat jiný způsob nastavení poloměru. Jeden nový způsob je v práci navržen (kapitola 4.1) a pomocí jednoduchých experimentů bylo ověřeno, že jej lze použít v praktických aplikacích.

2. *Rozšíření SDM.* Základní SDM má několik nevýhod: (1) Algoritmus inicializace nás nutí generovat rovnoměrně podmnožinu z celého adresového prostoru, i když při práci s konkrétními daty využijeme jenom malou část. Z toho vychází myšlenka na modifikaci algoritmu generování. Nové algoritmy jsou popsány a analyzovány v kapitole 4.2.1. (2) Původní SDM pracuje pouze s vektory bitů. Rozšíření na vektory reálných čísel lze realizovat podle návrhu v kapitole 4.2.3.

3. *Využití SDM při konstrukci složitějších sítí.* Hierarchická síť, navržená v kapitole 4.3.3, je inspirována Neocognitronem [4, 5] a prokázala schopnost rozpoznávat deformované číslice.