A New Approach to Kanerva's Sparse Distributed Memory

Tim A. Hely, David J. Willshaw, and Gillian M. Hayes

Abstract— **The Sparse Distributed Memory (SDM)[1] was originally developed to tackle the problem of storing large binary data patterns. The model succeeded well in storing random input data. However its efficiency, particularly in handling non–random data, was poor. In its original form it is a static and inflexible system. Most of the recent work on the SDM has concentrated on improving the efficiency of a modified form of the SDM introduced by Prager [2], which treats the memory as a single-layer neural network.(See also [3], [4], [5], [6].) This paper introduces an alternative SDM, the SDM Signal model [7] which retains the essential characteristics of the original SDM, whilst providing the memory with a greater scope for plasticity and self-evolution. By removing many of the problematic features of the original SDM the new model is not as dependent upon *a priori* input values. This gives it an increased robustness to learn either random or correlated input patterns. The improvements in this new SDM signal model should be also of benefit to modified SDM neural network models.**

Keywords— **SDM,Kanerva,memory,model,signal.**

I. Introduction

IN 1984, Pentti Kanerva introduced the Sparse Distributed Memory, or SDM, a new approach to tackling the problem of modelling human memory [1]. Kanerva's model showed some architectural similarity to the structure of the cerebellum and succeeded quite well in storing random input data. However, its efficiency, particularly in handling non–random data, was poor. We present a development of the SDM model which is very efficient and more flexible in dealing with different types of input.

The SDM consists of a large number of memory locations whose addresses are given by bit strings/binary patterns of length n, randomly distributed throughout the address space $\{0,1\}^n$. Figure 1 shows a simplified SDM where n=6. Each location is provided with a register whose 'bit'-positions are signed integer counters in which to store data. Input to the memory consists of two binary patterns, an address pattern with which a particular memory location is accessed, and a data pattern to be stored there. When writing data to the SDM, the address pattern is compared against all memory location addresses. The number of different bits between two binary patterns is known as the *Hamming distance*. If the Hamming distance between input address and memory address is less than a pre-specified amount, the *select radius*, a copy of the input data pattern is added to the contents of the data counters at that location. A '1' in the input data pattern increases by 1 the value of the counter at the corresponding position in the register; a '0' decreases the value by 1. With random data (P(1)=P(0)=0.5), the average counter value will be zero. However with non–random data (e.g.P(1)=0.75,P(0)=0.25) the average counter value will be positive if P(1)>P(0) and negative if P(0)>P(1). Recently [5] have shown that implementing a variable threshold based on the average number of 1's and 0's presented to the memory significantly improves the performance when handling non–random data. In this paper, a threshold fixed at 0 was used with random data and non–random addresses.

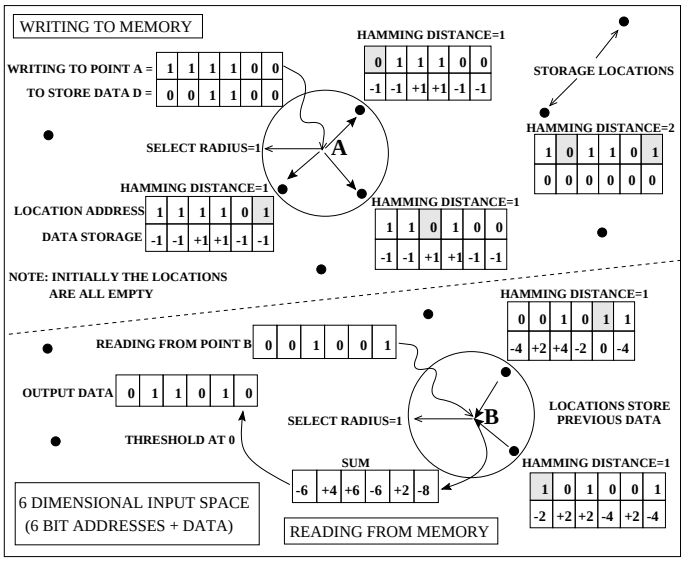

Fig. 1. Visualising the SDM with 6 bit addresses and data. Select radius=1. Shaded boxes show location address bits which differ from the input address. **Top:** 3 locations are selected in writing to the memory at point A. The location with Hamming distance 2 is not written to. The Hamming distance has not been shown for all locations. **Bottom:** Reading from point B selects 2 locations which have previously been written to. Their storage counters are summed and thresholded at zero to give the output data.

In Figure 1 the data is stored at three locations which have not previously been written to. Writing to the memory stores input data at locations whose addresses are most similar to the input address. When reading from the memory, all locations having a Hamming distance less than the select radius are accessed. In the reading operation in Figure 1, two locations storing previous data are selected. In the original SDM, the contents of the counters at those locations are summed and thresholded at zero to give a binary output pattern. Reading from any point in the memory space thus pools the data of all nearby locations.

II. Addressing the Shortcomings of Kanerva's SDM

The postulates underlying the original SDM are [8, page 54]:

T. Hely and D. Willshaw are at the Centre for Cognitive Science, The University of Edinburgh, 2 Buccleuch Pl., Edinburgh, Scotland, UK, EH8 9LW: email (tim,david)@cns.ed.ac.uk

G. Hayes is at the Department of Artificial Intelligence, The University of Edinburgh, 5 Forrest Hill, Edinburgh, Scotland, UK, EH1 2QL: email gmh@aifh.ed.ac.uk

1. The storage locations and their addresses are given from the start, and only the contents of the locations are modifiable.
2. The storage locations are very few in comparison with 2^n (the memory is sparse.)
3. The storage locations are distributed randomly in the $\{0,1\}^n$ address space.

The memory which results from postulates (1) and (3) is inflexible to both non–random and changing input addresses. In the original SDM, data is stored at all locations lying within the select radius of the input address, and each location receives a unit copy of the data signal. For a binary word 256 bits long, the best value of this select radius (the maximum Hamming distance allowed) which creates an efficient SDM is fixed at 110. With a memory of size 1000 locations, this selects about 10 locations each time (approximately 1%), and provides the best efficiency. However, if either the select radius cannot be chosen so accurately in advance, or the input pattern length varies, the performance of the memory rapidly deteriorates. Decreasing the input pattern length from 256 to 128 bits selects 100% of locations, whilst an increase to 512 bits selects 0% of locations. (This can be calculated from binary probabilities of obtaining <110 when N=128 or <110 when N=512, given that P(1)=P(0)=0.5.) As seen later in the Tests section, all the above limitations severely restrict the use of the standard SDM,

To achieve our aims of improving the performance and flexibility of the SDM, it was necessary to modify postulates 1 and 3 of Kanerva's original SDM, and introduce a fourth postulate. The new postulates are:

1. The storage locations that make up the final memory are not known at the start. Initially locations are created until there is an excess of storage locations which then compete for the available signal. Storage locations receiving little or no signal are removed. Locations which survive are chosen for the total amount of signal they receive.
2. The storage locations are very few in comparison with 2^n (the memory is sparse).
3. Although the storage locations are initially distributed randomly in the $\{0,1\}^n$ address space, the final distribution of locations depends upon the input patterns presented, and may be non–random.
4. The select radius (read/write radius) of the original SDM is replaced by a new parameter which decreases the value of the signal as it spreads out. Locations have real valued counters to store a copy of the data weighted by the strength of the signal they receive. The signal does not propagate after it falls below a minimum strength.

The new Postulates 1 and 3 allow the addresses of locations to have a non–random distribution if the input addresses are non–random. If however the input is random, the locations are randomly distributed as before, corresponding to Kanerva's SDM. Figure 2 illustrates the new signal loss model to which Postulate 4 refers.

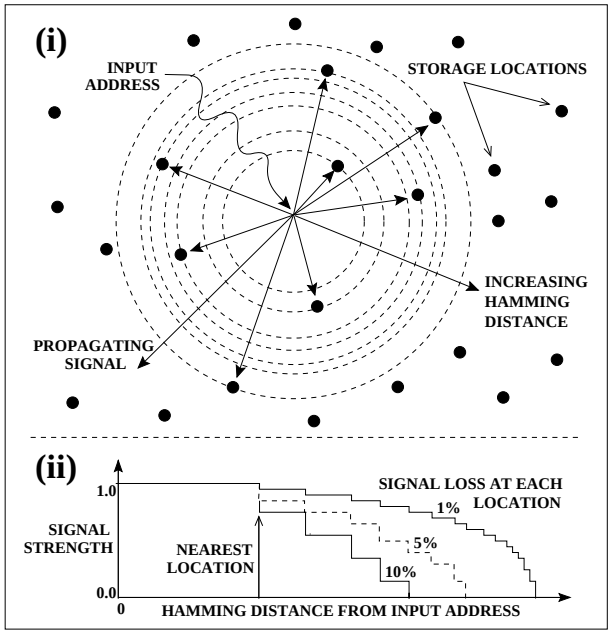

Fig. 2. Mechanism of signal loss in the new model. As the signal travels outwards it loses a small percentage of its momentum at each location.

The data pattern is first stored at locations which lie closest to the input address. The signal (i.e data pattern) then spreads throughout the memory, and a small percentage of the signal strength (e.g. 5%) is lost at each subsequent location encountered. If the signal was propagated throughout the whole memory, all locations would receive a copy of every input pattern, and the memory would soon be saturated. Distributing the signal in this way removes the need for a select read/write radius, one of the problematic features of the original SDM. All locations selected in a write operation do not now receive a copy of the original binary pattern with equal strength. Instead they receive a copy of the pattern weighted with a real value from 1.0→0.05 to store in real valued counters (rather than binary counters in Kanerva's SDM). This rewards the nearest locations with a greater signal strength, and uses the natural architecture of the SDM to attenuate the signal strength. Similarly in reading from the memory, output from the nearest locations is given a greater weight than from more distant locations. The new Signal model is flexible to varying input as the loss factor does not have to be changed for input patterns of different lengths.

III. The New SDM Signal Model

In the original SDM, locations were randomly distributed in the input space. In the initial set–up phase of the new model, memory locations are created at points in the input space where signal is being received, e.g. at active input sites. The address of a new location is then given the identical address to the input pattern being presented. In this way, the memory locations of the new SDM model can map directly onto any input patterns which are presented. Data is not stored until all locations have been set up.

During this initial phase, memory locations are created until there is an excess, each location then being in competition for the available signal. Figure 3 plots the number of 'alive' locations against the number of patterns presented. The new signal method allows the total signal strength received by a location to be used as a measure of the fitness of a location. Once the setting up phase is complete, the trial period consists of writing 100 patterns to the memory. In the killing period, a further 100 patterns are written to the memory. During this period, locations which have received the least signal are killed off at regular intervals until, after 200 patterns, only 1000 locations remain. The new SDM is made up of only the 'fit' locations which survive. These locations then have their counters reset before the 300 test patterns are presented.

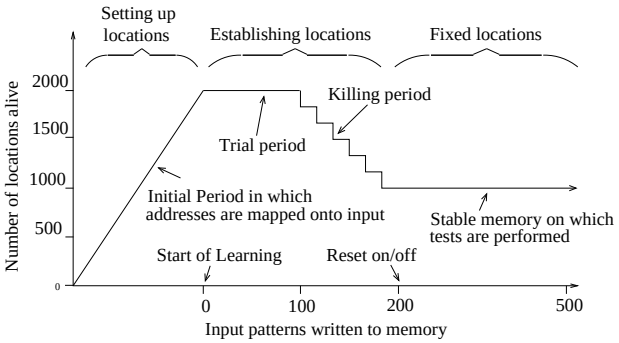

Fig. 3. Setting up the new signal model. Each location has a 256 bit address and data counters. Of the initial 2000 locations, 1000 locations are killed off after the trial period by removing those locations storing the fewest patterns. The remaining 1000 locations are then used in the subsequent tests.

This pruning process was not intended to find the best architecture mathematically (See [3]), but primarily to create a memory with 'fit' locations, which was of equal size to the modified SDM of David Rogers [9]. This allows a direct comparison to be made in efficiency tests between the new SDM Signal model and previous SDM models. These tests are described in the following section.

IV. Tests

A. Random Data Tests

Rogers [9] created a model of the SDM, known as a data-tagging model which differs chiefly from the original SDM in its reading operation. In the data-tagging model, the output pattern is the pattern which has been written *most frequently* to the selected locations, and is *not* a pooled average of the data stored in the counters. This requires a pattern bank in which to store and count previously presented patterns. This version of the SDM has an improved efficiency. However, as the number of data patterns increases (particularly above 8000 locations), it becomes very expensive to allocate the storage needed.

Rogers also developed a number of tests to allow an efficiency comparison of different SDMs. All of the tests first write a data pattern D to an arbitrary point A in an empty memory (see Figure 1), and then monitor how that initial data is corrupted as additional patterns are presented. In the standard tests, random addresses and data patterns are used. As the memory fills up, random input gradually overwrites the data originally written at A, and errors occur in the output. The results for the standard SDM, Rogers' data–tagging SDM, and the new Signal model are given in Section 5.

B. Correlated Tests

The original SDM performs best when random binary words are used as input addresses, i.e. when P(0)=P(1)=0.5. The correlated test determines the performance of the SDM when the input address patterns are non–random (the data pattern remains random). The word 'correlated' is used in the same context as in [9] to indicate a non–random effect, i.e. P(0)≠P(1)≠0.5. The *Q factor* is the total number of 1's in any given input address[1]. In the original SDM the performance is reduced dramatically when input patterns with either a high Q factor (more 1's than 0's) or a low Q factor (more 0's than 1's) are presented to the memory.

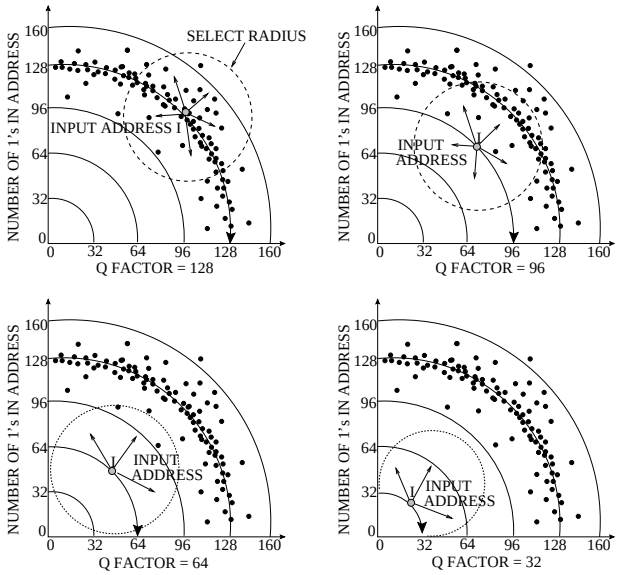

Fig. 4. Schematic 2–D representation of introducing correlated addresses into Kanerva's SDM. Axes show the Q factor (the number of 1's in an address pattern) of both the input address (shaded dot), and the location addresses (black dots). As the Q factor is reduced from 128 to 32, the region of memory space in which locations are chosen becomes less populated. See Section IV-B

Figure 4 shows a 2–D representation of the memory space. Given P(0)=P(1)=0.5, a 256 bit pattern is expected to have 128 1's and 128 0's. The addresses of most locations are clustered around the 128 contour line, and on average only 1% will have fewer than 107 1's or more than 149[2]. In the correlated tests the input address patterns contain only 96, 64 or 32 1's, which may occur at any of the 256 bit positions in the pattern. As shown in Figure 4, reducing the Q factor (increasing the correlation) forces

[1] N.B. A highly correlated pattern in [9] will in fact have a **low** Q factor.

[2] The distribution is binomial with p=q=0.5, mean=128 and standard deviation=7.

the memory to use only a fraction of its possible locations. The region where locations are selected is less populated. Fewer locations are chosen, and the locations selected will receive a copy of nearly every pattern. As a result, these locations quickly become saturated and the memory is soon corrupted.

V. Results

The new SDM Signal model has an equivalent performance to Kanerva's original SDM in random tests and a significantly improved performance in the correlated address tests. The model also outperforms Rogers' data–tagging model for address patterns with a high degree of correlation. Table I shows the average number of bits of the original 256 bit data string which are corrupted in a memory with 1000 locations after an extra 300 patterns have been stored. For Kanerva's and Rogers' SDM the select radius is set at 110. The signal loss is set at 10 percent for each location encountered in the Signal Model. 50 trials were used for Kanerva's SDM and the new Signal model. whilst results for Roger's model are based on 8 trials. (Standard deviation information was not available for Rogers' model.)

TABLE I
Mean error and standard deviation results for efficiency tests using different SDM models. See text for details

	Model		
	Kanerva	Rogers	Signal Model
Random Address Test	12.7(± 6.5)	8	15.4(± 7.2)
Q factor = 96 (or 160)	27.1(± 9.8)	10	16.6(± 6.5)
Q factor = 64 (or 192)	66.7(±12.1)	13	16.6(± 7.1)
Q factor = 32 (or 224)	102.7(± 8.3)	50	18.9(± 6.8)

Figure 5 shows the performance of the various models as the Q factor is decreased. Kanerva's SDM quickly deteriorates for non–random address patterns containing only 96 1's. At a Q factor of 32, the mean error (102 bits) approaches the expected error of 128 bits between two independent random patterns. (The maximum possible error is 256.) Rogers' data–tagging model performs well until the Q factor drops below 64, at which point the number of errors dramatically increases. However the new Signal model is able to create locations which map onto both random and non–random input address patterns, and the performance is stable even when all the input patterns are being stored in only a tiny fraction of the possible input space.

VI. Discussion

The original postulates governing Kanerva's SDM create a memory which performs poorly in dealing with non–random input addresses and data. Recently [5] have overcome some of the difficulties associated with storing and retrieving non–random input data. However the poor performance associated with non–random input addresses is largely due to the select radius which does not allow for any variation in the input. In our attempt to move away from a static memory entirely defined from its outset, we have created a new Signal model which overcomes the difficulties associated with choosing the select radius. A literature search did not find any instance of the new signal parameter having been previously used. ([1], [2], [3], [4], [5], [6], [8], [9], [10], [11], [12], [13])

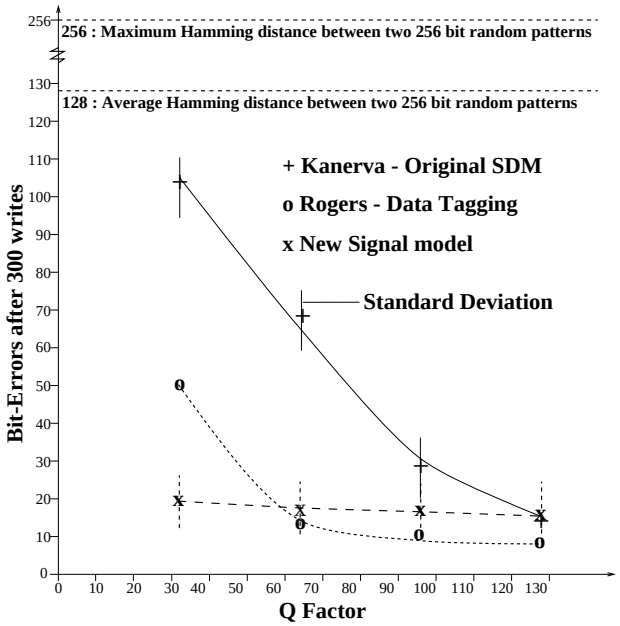

Fig. 5. Errors after storing 300 patterns v Q factor (No. of 1's in 256 bit pattern) using 1000 memory locations.

Both the method of drawing location addresses from a training set, and using a pruning technique to eliminate locations have been used separately before ([3], [14] respectively). However in both of these cases, the network developed has been restricted by the use of the inflexible select radius. The advantage of propagating the signal from its origin is to allow SDM locations to map directly onto the training set. A lengthy trial and error process to find the best select radius before the memory can be used is not required. The new Signal model also achieves a greatly improved efficiency when presented with non–random address patterns. It retains the sparse distribution of the original SDM, but replaces a static, inflexible model with a new powerful and dynamical system.

References

[1] Kanerva, P. (1984). Self-Propagating Search: A Unified Theory of Memory. Technical report, Research Institute of Advanced Computer Science.

[2] Prager, R. and Fallside, F. (1989). The modified Kanerva model for automatic speech recognition. *Computer Speech and Language,3, 61-81.*

[3] Prager, R. (1992). Some experiments with fixed non-linear mappings and single layer networks. Technical report, Cambridge University Engineering Department CUED/F-INFENG/TR.106.

[4] Baram, Y. (1994). Memorizing Binary Vector Sequences by a Sparsely Encoded Network. *IEEE Transactions on Neural Networks 5-6, 974–981.*

[5] Ryan, S. and Andreae, J. (1995). Improving the Performance of Kanerva's Associate Memory. *IEEE Transactions on Neural Networks 6-1, 125–130.*

[6] Holden, S. and Rayner, P. (1995). Generalization and PAC Learning: Some New Results for the Class of Generalized Single-Layer Networks. *IEEE Transactions on Neural Networks 6-2, 368–380.*

[7] Hely, T. (1994). The Sparse Distributed Memory: A Neurobiologically Plausible Memory Model? Master's thesis, Dept. of Artificial Intelligence, Edinburgh University.

[8] Kanerva, P. (1988b). *Sparse Distributed Memory.* MIT Press - a Bradford book.

[9] Rogers, D. (1988). Using data tagging to improve the performance of Kanerva's Sparse Distributed Memory. Technical report, Research Institute of Advanced Computer Science TR-88.1.

[10] Keeler, J. (1987). Capacity for patterns and sequences in Kanerva's SDM as compared to other associative memory models. Technical report, Research Institute of Advanced Computer Science TR 87.29.

[11] Kanerva, P. (1988a). The Organization of an Autonomous Learning System. Technical report, Research Institute of Advanced Computer Science TR 89.02.

[12] Rogers, D. (1989). Statistical Prediction with Kanerva's Sparse Distributed Memory. Technical report, Research Institute of Advanced Computer Science TR 89.02.

[13] Miles, C. and Rogers, D. (1993). A Biologically Motivated Associative Memory Architecture). Technical report, Dept. of Neurology, Baylor College of Medicine, Houston.

[14] Joglekar, U. (1989). Learning to read aloud: a neural network approach using Sparse Distributed Memory. Technical report, Research Institute of Advanced Computer Science 89.27.