An Adaptive Sparse Distributed Memory

J. L. Aguilar

CEMISID. Dpto. de Computación.
Facultad de Ingeniería. Universidad de los Andes.
Av. Tulio Febres. 5101. Mérida, Edo. Mérida-Venezuela
Telf: (58.74)440002 Fax:(58.74)402872
email: aguilar@ing.ula.ve

Abstract. **Sparse Distributed Memory is a content addressable, associative memory technique which relies on close memory items tending to be clustered together, with some abstraction and blurring of details. This paper discusses the limitations of the original model. Then, we propose a method which improve Sparse Distributed Memory efficiency through an adaptive threshold. The results obtained are good and promising.**

I. Introduction

The Sparse Distributed Memory (SDM) was developed by Pentti Kanerva [4], and it may be regarded either as an extension of a classical or as a special type of three layer feedforward neural network. We can simulate some of the human cognitive capabilities because it work in a similar way of the human memory (associative memory recognition, etc.). SDM implements transformation from logical space to physical space using distributed data storing. A value corresponding to a logical address is stored into many physical addresses. This way of storing is robust and not deterministic. In, general, the main properties are [4]: a physical address can keep several logical addresses, the memory capabilities are large, a memory cell is not addressed directly, and if logical addresses are partially damaged, we can get correct output data.

The SDM standard uses a fix threshold to active the memory cell [1, 2, 5]. That simplifies the system but gives poor results because some of the memory cells can't be reusable due to that a fix threshold only allow recovery the information previously learn. That limits the capabilities and fault tolerance of the system and some information storing can't be recovery (specifically, when the number of information store is large). In this paper we propose an adaptive threshold to solve this problem. We present some experiments with results, which are promising.

II. Sparse Distributed Memory

SDM is a model proposed by Karneva that can work as an auto-associative memory technique where the contents and the addresses are from the same space and used alternatively [4]. The inner workings of SDM rely on large binary spaces. The dimension of the space determines how rich is each word. Another important factor is how many actual memory locations are there in the space. In general, when we use a SDM, the features are represented as one or more bits. Groups of features are concatenated to form a word which becomes a candidate for writing into SDM. When writing copy of this binary string is placed in all close enough hard locations. When reading, a close enough cue would reach all close enough hard locations and get some sort of aggregate or average out of them. Reading is not always successful. Depending on the cue and the previously written information, among other factors, convergence or divergence during a reading may occur. If convergence occurs, the pooled word will be the closest match of the input reading cue. On the other hand, when divergence occurs, there is not relation-in general-between the input cue and what is retrieved from memory. The main aspects of SDM are [1, 2, 3, 4]:

- The SDM calculates Hamming distances between the reference address and each location address. For each distance, which is less or equal to a given radius, the corresponding location is selected.
- The memory is represented by n*m counters (where n is number of locations and m is the input data length) instead of single-bit storage elements.
- Writing to the memory, instead of overwriting, is as follows:
  - If the i-bit of the input data is 1, the corresponding counters (counters in the selected locations (rows) and in the i-th columns) are incremented
  - If the i-bit of the input data is 0, the corresponding counters are decreased
- Reading (or recall) from the memory is similar:

0-7803-7898-9/03/$17.00 ©2003 IEEE

- The contents of the selected locations are summed columnwise.
- Each sum is thresholded. If the sum is greater that or equal to the threshold value, the corresponding output bit is set to 1, in the opposite case it is cleared.

Some of the limitations of the SDM are [2, 3, 5]:

- The addresses and the data vector must be coded in a binary alphabet. Because a lot of real problems using a different alphabet, we need a system to translate the pattern of real problems in a code to be used for the SDM.
- The standard model uses an uniform distribution at the level of the input addresses. In the reality, the input vectors are grouped in a set distributed in a large multidimensional space. For that, if the addresses are choose randomly like is proposed by Karneva, a large number of cells maybe can't activate and other ones would be activate a lot of time.
- If we save a lot of data in a SDM, we can overlap a lot of information. In this case the results of the output vector will be incorrect. Only, if we store several time the same information, the probability to recovery a correct information that we have stored can increase. The reason is because the standard SDM use a fix threshold to activate the cells. That gives poor performances. For this last case is that we propose our approach.

III. Our Adaptive Threshold Approach

The standard SDM has been modified in order to use an adaptive threshold. That is, the decodification of the addresses is not fixed, and the threshold is modified according to the results of the recognition phase of previous learned patterns. If the recognition rate is bigger than 95% that means that the threshold must continue fix, otherwise we must decrease or increase the threshold according to the quality of the recognition of previous learned patterns and the number of input patterns. Once modified the threshold, the SDM relearns the set of pattern with the new threshold. In this way, we can filter the information and introduce them in the SDM in the correct places. The threshold will be modified according to the next conditions:

1. If the recognition rate (Tr) is bigger than 95% then:

$$threshold(t) = threshold(t-1) \quad (1)$$

For this case, the threshold is not modified for the recognition phase.

2. If the recognition rate (Tr) is smaller than 95% and the data quantity stored is smaller than (number of memory cells/2) then:

$$threshold(t) = threshold(t-1) - (100 - Tr) * \bullet \quad (2)$$

- Where Tr is the recognition rate, which is calculated during the recognition phase as the average recognition rate:

$$Tr = \sum_{i=1}^{N} Tr(i)/N$$

- $\bullet$: is the learning rate ($0 \leq \bullet \leq 1$). In our experiment we have used $\bullet = 0.1$ because with it we have obtained the best results, see [2] for more details.
- N: number of information to be recovered.

For this case, the initial threshold for the learning phase must be a large value. In our experiment, we use 5 (see [2] for more details).

3. If the recognition rate (Tr) is smaller than 95%, and the data quantity stored is bigger than (number of memory cells/ 2) then:

$$threshold(t) = threshold(t-1) + (100 - Tr) * \bullet \quad (3)$$

- For this case, the initial threshold of the learning phase must be a small value. In our experiment, we use 3 (see [2] for more details).

The equations (1) to (3) are used iteratively until to improve the recognition rate *(Tr)* of the previous learned pattern. Then, we restart the learning phase of the SDM with the new threshold.

IV. Experiments

In our experiment we use an Applet in Java to simulate the SDM. In addition, we compare our results with [1, 4]. We use the next set of parameters: the size of the input and output binary vectors of the SDM, the number of memory cells (NLA), initial threshold and the number of input patterns (N).

A. First Experiment

In this experiment, NLA is 15, N is 5 and the size of the input and output binary vectors of the SDM is 10 bits. During the learning phase, we learned the next set of information with threshold=5: {0111110000, 0011111000, 0001111100, 0000111110, 1000001111}. The recognition rate of these learned patterns, with threshold=5, is equal to 83% (see [2]).

Because Tr = 83 % ≤ 95 %, we recalculate the threshold according to the equation (2),

$$\text{Threshold}(t) = 5 - (100-83)*0.1 = 3{,}3 \approx 3$$

In this case, the recognition rate, with threshold = 3, is equal to 88%. Because Tr = 88 % ≤ 95 %, we recalculate the threshold according to the equation (2),

$$\text{Threshold}(t) = 3 - (100-88)*0.1 = 1{,}8 \approx 2$$

Now the recognition rate, with threshold = 2, is equal to 83%. Because Tr = 83 % < 88 % with threshold = 3, the optimal value of the threshold is 3. We relearn the SDM with the threshold=3 and we obtain a recognition rate of the previous learned patterns equals to 95%. If we introduce some noise during the recognition phase, we obtain the next results:

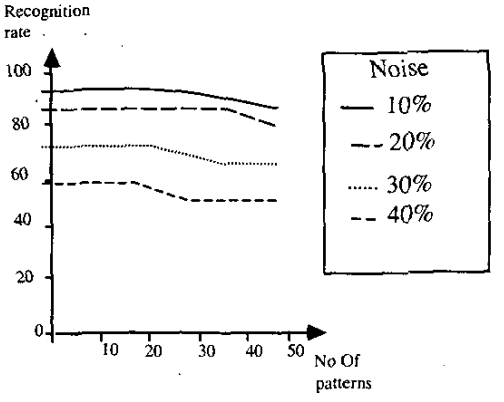

Fig. 1. Recognition rate Vs noise

Now, in this phase we compare the performance of our approach with [1] and [4]. For each case, we have developed 30 experiments.

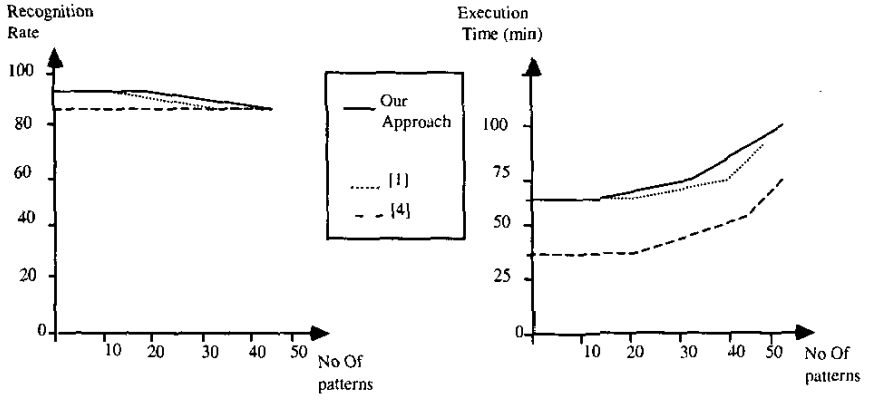

Fig. 2. Comparison of the Recognition rate and execution time for a noise = 10%

We improve the results obtained in [1, 4]. The reason is because our approach can adapt its procedure to the quality of the figures using the adaptive threshold. The execution time is very close to the evolutionary approach.

B. Second Experiment:

With a number of input patterns (N) = 10, we use the next set of data during the learning or written phase with an initial threshold = 3: {0111110000, 0011111000, 0001111100, 0000111110, 0000011111, 1000001111, 1100000111, 1110000011, 1111000001, 1111100011}. The recognition rate of previous learned patterns, with threshold=3, is equal to 91% (see [2]). Because Tr = 91 % ≤ 95 %, we recalculate the threshold using equation (3),

$$\text{threshold}(t) = 3 + (100-91)*0.1 = 3{,}9 \approx 4$$

In this case, the recognition rate, with threshold = 4, is equal to 92%. Because Tr = 92 % ≤ 95 %, we recalculates the threshold using the equation (3),

$$threshold(t) = 4 + (100-92)*0.1 = 4,8 \approx 5$$

In this case, the recognition rate, with threshold = 5, is equal to 86%. Because Tr = 86 % < 92 % with threshold = 4, the optimal value of the threshold is 4. We relearn the SDM with the threshold=4 and we obtain a recognition rate of the previous learned patterns equals to 93%. If we introduce some noise during the recognition phase, we obtain the next results:

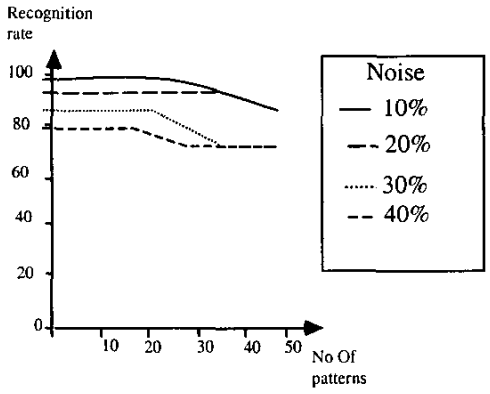

Fig. 3. Recognition rate Vs noise

Now, in this phase we compare the performance of our approach with [1] and [4].

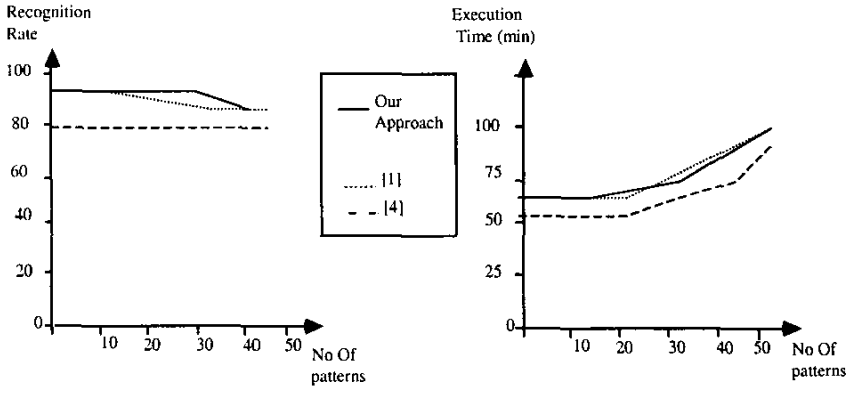

Fig. 4. Comparison of the Recognition rate and execution time for a noise = 10%

Similar to previous results, our approach improves the performance of previous work because we have obtained the optimal value of the threshold.

C. Third Experiment

In this part, we compare the performance of our approach for different number of bits of the input patterns. We suppose NLA=15 and N=10.

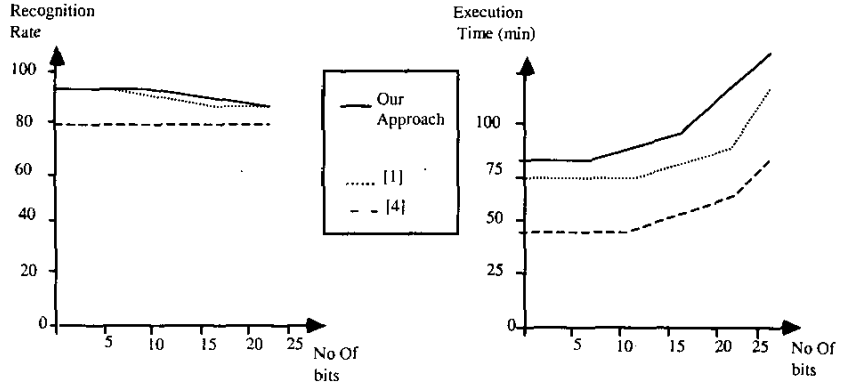

Fig. 5. Comparison of the Recognition rate and execution time for a noise = 10% and different number of bits of the input patterns

In this case, the pattern size doesn't affect the performance of our approach. The problem is with our execution time, this increases when the number of bits is large.

V. CONCLUSIONS

The experimental results show that if we use an adaptive threshold in the SDM we can improve the performance of the recognition phase with respect to the classical SDM approach. In general, we obtain similar performance than the evolutionary approach proposed in [1]. But the recognition rate is the best for large N (one of the main capabilities of the SDM approaches is that). The execution time is increase because we relearn the SDM with the optimal value of threshold. We are going to extend our experiments for more complex problems (images recognition with a large number of bits, etc.).

REFERENCES

[1] J. Aguilar, A. Colmenares, "Resolution of Pattern Recognition Problems using a Hybrid Genetic/Random Neural Network Learning Algorithm", *Pattern Analysis and Applications*, vol. 1, pp. 52-61, 1998.

[2] J. Aguilar, "Some experiments on the Sparse Distributed Memory", Technical Report 22-02, CEMISID, Universidad de los Andes, 2002.

[3] F. Grebenicek, "Data coding for Sparse Distributed Memory". *Mosis'98 Proceedings*, pp. 34-40, 1998.

[4] F. Grebenicek, "Self-Organized Sparse Distributed Memory- an Application: Pattern Data analysis". Technical Report 124, Ostrava University, 2000.

[5] P. Kanerva, *Sparse Distributed Memory*. The MIT Press, Cambridge, Massachussetts, 1990.