This is a pre-copyedited, author-produced version of an article accepted for publication in Lecture Notes on Data Engineering and Communications Technologies following peer review. The final authenticated version is available online at https://doi.org/10.1007/978-3-030-70665-4_107

Integer Sparse Distributed Memory Based On Neural Coding

Xiaowei Wang and Hongying Meng

Brunel University London, Kingston Lane Uxbridge Middlesex UB8 3PH, UK
{xiaowei.wang,hongying.meng}@brunel.ac.uk

Abstract. Sparse Distributed Memory (SDM) is a mathematical associative human long-term memory model that is psychologically and neuroscientifically plausible. It is content addressable and can potentially store unlimited high dimensional information although it only accepts binary vectors. Although an integer SDM has been proposed to store non-binary vectors, the size of the model has been increased significantly. In this paper, we propose a biologically plausible encoding schema to enable SDM to accept integer vectors, requiring much less space than the existing integer SDM model. A public handwritten digits data set with integer values has been used for the evaluation of our proposed model and satisfied performance has been achieved in both data retrieval and pattern recognition.

Keywords: SDM, sparse distributed memory, neural coding

1 Introduction

The Sparse Distributed Memory model (SDM) was introduced in 1988 by Kanerva [1]. Its purpose is to simulate human long-term memory which has an potential infinity information storage mechanism although the size of the memory is limited by the fixed number of hard locations. SDM is distributed, auto associative, content addressable, and noise robust. It represents a mapping from concepts in human mind to n-dimensional space.

The original SDM, due to its structure design with binary address vectors, has a fundamental limitation. It only accepts binary data directly, while data in real life is not always binary. Various encoding schemes were evaluated by Mendes et al. [2], but none of them were capable of storing a non-binary value type in a single dimension in SDM. A significant improvement of SDM was made by Snaider et al. [3]. They proposed an integer SDM that extended SDM to store integer values in a single dimension of the memory. However, storage and computing costs increased significantly. It also increased the complexity of model hardware implementation.

Here we explored a new methodology that allows SDM to process integers using a neural coding scheme. The rest of this paper is organized as follows: Section 2 introduces SDM related works. Section 3 explains our methodology. Some simulation results are presented in Section 4. Section 5 concludes the paper.

2 Related Works

In the past 30 years, SDM has been improved with various extensions increasing its performance and its applications. Hard location activation was made faster by [4] and [5]. Hely et al. [6] proposed a SDM signal model that didn't rely on *a priori* input values, and it increased the robustness of SDM learning both random and correlated data. Ternary memory space was introduced by [7]. This improved SDM performance as the percentage of missing features in the read-cues increases to a reasonable degree. In 2009, Meng et al. [8] proposed a modified SDM model that initialized the address matrix using training samples in a straightforward fashion which significantly improved recall performance. They also introduced a tri-state technique in learning rules to keep the storage requirement low while increasing the performance significantly.

The integer SDM model proposed by Snaider et al. [3] in 2012 has a similar structure to Kanerva's binary SDM. But the values in the address matrix are integers within a value range. The size of the range is r, and there is no limitation on the value of r. The dimensions of the space follow modular arithmetic, i.e. the values wrap around after r. Instead of holding only one counter for each cell, each dimension of a vector in the content matrix holds r counters. The procedures of reading and writing are similar to the ones used by SDM. Both writing and reading operations need to find activated hard locations first. When writing a word, only one counter in each dimension of the activated hard location can increment out of the total r counters. By comparing the value in the same dimension of the written word and the value of each counter in that dimension, the counter increments if its index is equal to the value in the word. To read from memory, the counters of each dimension with the same index in the activated hard locations are summed up. The majority rule is applied to the collection of sums and the index of the counters with the largest sum is the output value of that dimension of the output vector. Snaider's integer SDM model has a space complexity $\mathcal{O}(mnr)$ while the original SDM model has space complexity $\mathcal{O}(mn)$. Its time complexity is $\mathcal{O}(mn + prmn)$ compared to $\mathcal{O}(mn + pmn)$.

In 2018, Huang et al. [9] proposed a spiking pattern classifier. Although the work didn't involve SDM, the way to transform spikes into a high dimensional sparse vector using a temporal encoding still provides useful insights for our work.

3 Methodology

3.1 Overview

To extend the original SDM model to accept integer vectors, a neural coding scheme was proposed, instead of modifying the SDM model's internal structure. Figure 1 shows an overview of our method. The original image whose pixel values are integer is transformed into an integer vector as an input of the SDM model. The spiking neural encoder converts each integer value in the vector into a sequence of binary data. Therefore, the integer vector is converted into

a sequence of binary vectors as shown in Figure 2, before it is written into the SDM. When reading from the SDM model, similar to writing operation, the integer vector cue is encoded into a sequence binary vector cues first. The output sequence of binary vectors from SDM is converted back into a single integer vector using the spike neural decoder as shown in Figure 2. The integer vector is then transformed back into an integer pixel matrix, the recalled image.

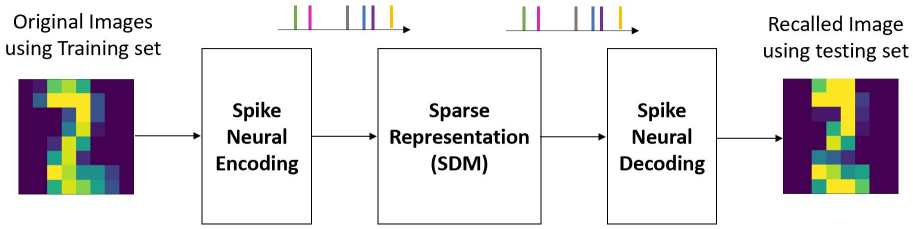

Fig. 1. Overview of proposed integer SDM model with neural rate coding.

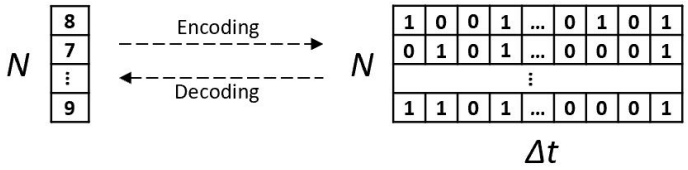

Fig. 2. The transformation between Integer vector and sequences of binary vectors using neural encoding and decoding.

3.2 Neural Coding

Different neural coding schemes have their own merits and are based on biological observations [10]. The coding scheme we chose here was inspired by the spike train generation, which is observed from *in vitro* experiments in neuroscience. A spike train is a sequence of recorded times at which a neuron fires an action potential. When the voltage drop across a neural soma or axon membrane is recorded, intermittent pulses of roughly 100 millivolts over 1-2 milliseconds are observed—these are action potentials or spikes. On a behavioral time scale of several hundred milliseconds, each spike may be considered to occur at a single point in time. Sequences of such spike times form spike trains. Spike trains are considered to be the primary mode of information transmission in the nervous system. When action potentials are recorded repeatedly from a neuron in response to changing stimuli, they maintain a relatively consistent shape. However, the pattern of spike times varies as the stimulus changes. Most prominently,

the rate at which spikes occur, the firing rate (f), varies with conditions. When the firing rate is defined in terms of the maximum number of spikes (L) that occur over a time interval of length Δt, using

$$f = \frac{L}{\Delta t} \tag{1}$$

this phenomenon of firing rate varying with stimulus is called rate coding. While analysis of neural activity in terms of varying firing rates, as defined in Equation 1 which represents spike count rate, is useful in many contexts, more subtle alterations of the pattern of spike times also occur. Here, we used a spike count rate coding scheme for encoding and decoding between an integer value and its spike train. For an integer value $I \in [0, K]$, it will be converted into a binary sequence $\{b_1, b_2, \cdots, b_{\Delta t}\}$ satisfying

$$\sum_{k=1}^{\Delta t} b_k = I \times \frac{L}{K} \tag{2}$$

where $b_i = 1 (i = 1, 2, \cdots, \Delta t)$ means a spike and $b_i = 0$ means no spike. For one integer value, its binary sequence can be chosen differently. But all these binary sequences will be decoded to the same integer.

3.3 Sparse Distributed Memory

The Kanerva's SDM model [1] as shown in Figure 3, contains two matrices: an $N \times M$ address matrix $\mathbf{A}$ which contains binary data, and a $U \times M$ content matrix $\mathbf{C}$ that contains integer data. It is content-addressable (i.e., an input word $\mathbf{w}$ is also used as an address $\mathbf{x}$), so the column widths of the matrix $\mathbf{A}$ and matrix $\mathbf{C}$ are the same, $N = U$. Instead of listing all addresses in memory space, matrix $\mathbf{A}$ keeps a small portion of sparse randomly distributed addresses, called hard locations. A hard location contains an address vector in matrix $\mathbf{A}$ and an integer vector in matrix $\mathbf{C}$. When writing or reading data, a set of hard locations within a activation radius $\mathbf{r}$ are activated after comparing the hamming distance between the input address $\mathbf{x}$ and each hard location address. When matrix $\mathbf{A}$ holds a uniform random sample of address space, the binomial distribution with parameter N can be used to find the activation radius $\mathbf{r}$ that corresponds to a given probability p of activating hard locations, according to Equation 3, from [11],

$$r = \mu - 3 * \delta \tag{3}$$

where $\mu = \frac{N}{2}$, $\delta = \sqrt{\frac{N}{4}}$. The values in the distance vector $\mathbf{d}$ are hamming distances and the 1s in the activation vector y mark the activated hard locations. For writing operation, at an activated hard location, each dimension of the integer content vector will increment by 1 if the value of the same dimension of the input word is 1, or decrement by 1 if the value is 0. For reading operation, a dimension in the output vector $\mathbf{z}$ is set to 1 when the sum is larger than 0, or 0 otherwise, after summing each dimension of the content vectors into a vector $\mathbf{s}$ at the activated hard locations.

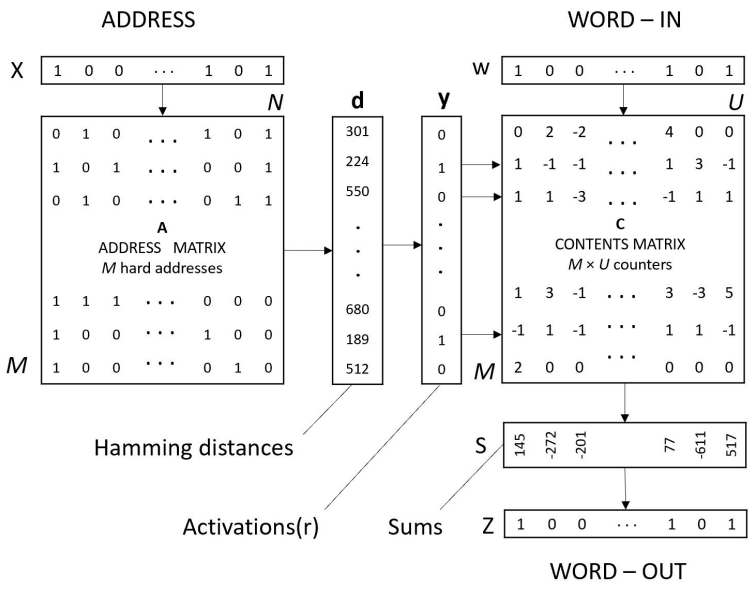

Fig. 3. Structure of Binary Sparse Distributed Memory Model.

4 Evaluation

4.1 Data Set

An optical recognition of a handwritten digits data set [12] was chosen for the evaluation of our integer SDM model. This data set contains 1797 images with labels ranging from 0 to 9 and a resolution of 8x8 pixels. The range of each pixel value is from 0 to 16. A SDM model will be built with 1257 images for training the model and 540 images for testing the performance on data retrieval/recall and pattern recognition using a simple classifier.

4.2 Parameter setting

Neural encoder/decoder parameters: Sample training time window Δt: defines how many binary vectors to convert from an integer vector. There is no value limitation, but we set it to 16.

The number of maximum spikes L affects how sparse of the spike train is. Here, 8, 12, and 16 are chosen for the experiments. $L = 16$ means the fire rate $f = 1$ and $L = 8$ means the fire rate $f = 0.5$.

SDM parameters: The address matrix was initialised with a sparse randomly and uniformly distributed (bits were independent of each other, and the ratio of 0s to 1s was around 1.0) sample of addresses and the content matrix was initialised with 0s.

Memory space dimension is chosen as $N = 64$ as an 8x8 image can be transformed into a 1x64 integer vector. $N = 64$ for both address and content matrices. Activation radius of writing $r_{write} = 20$. The SDM reading radius $r_{read} = 25$.

Hard location amount M: This number stands for the capacity of the memory and the memory should be more powerful if this number is bigger. We tried 1000, 2000, 4000, 8000, 16000 and 32000 in our experiments.

4.3 Data Retrieval

This was to verify the data retrieval performance of the neural coding integer SDM as an auto-associative memory. In the data set, the first 1257 images were used as training set while the rest 540 images were used as testing set for retrieval. 100 of retrieval images are shown in Figure 4.

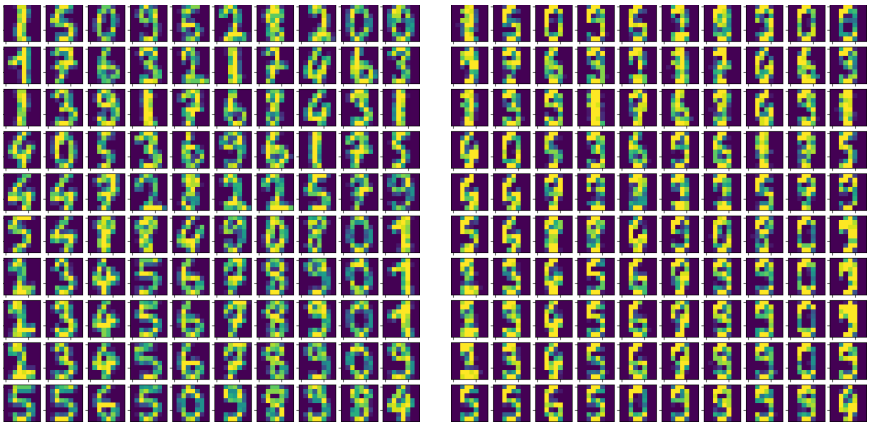

Fig. 4. 100 original (left) and recalled (right) images, $N = 64$, $M = 8000$, $r_{read} = 25$, $r_{write} = 20$, $\Delta t = 16$, $L = 16$.

4.4 Pattern Recognition

This experiment was designed to verify whether the changes to the stored integers were consistent in memory, compared to their original values. If the original data set was classifiable, the reconstructed images should still be classifiable if the retrieval quality is good. After the training using 1257 images, the SDM model is fixed. Then the 1257 reconstructed images from training set are used as the training set and the remaining 540 reconstructed images as testing set to train a Support Vector Machine (SVM) [13] classifier. The classification accuracy using radius basis function kernel (RBF), the regularization parameter $c = 1.0$ and $\gamma = 0.001$ are shown in Table 1 under different parameters. The

best performance is 94% compared to 97% using the same classifier on the original data set. It can be seen that with increase of the number of hard locations, the performance is better. With the increase of maximum spikes per pixel, the performance is also better. The classification confusion matrix under condition of $M = 8000, L = 16$ is shown in Figure 5 and compared to classification results on original data.

Table 1. Recognition accuracy under different settings when $r_{read} = 25$ and $r_{write} = 20$.

Number of hard location (M)	1000	2000	4000	8000	8000	8000	16000	32000
Max spikes per pixel (L)	16	16	16	8	12	16	16	16
Recognition accuracy (%)	86	86	92	76	91	93	93	**94**

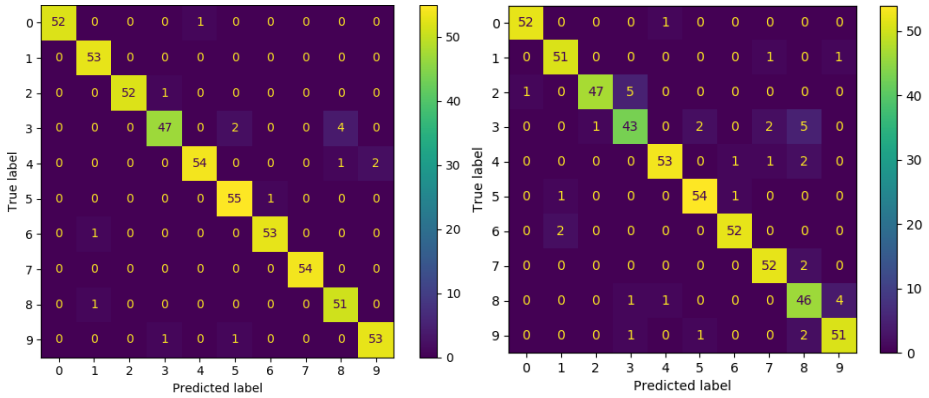

Fig. 5. Classification confusion matrix using original data set with accuracy of 97% (left) and using recalled data set from SDM with accuracy of 94% (right).

Assuming an integer requires 8 bits to be represented, the total storage space of SDM in this experiment is $M * N + M * N * 8 = 9MN$ bits. Using Snaider's integer SDM model, with the same value range size 17 (0–16) and the same amount hard locations, it requires $M * N * 8 + M * N * 8 * 17 = 144MN$, which is 16 times larger than our SDM implementation. It is a significant advantage. For every single integer datum stored, our method requires $M*N*\Delta t+M*N*p*\Delta t = M * N * 16 + M * N * 0.001 * 16 = 16.016MN$ operations, while integer SDM requires $M * N + M * N * p * r = M * N + M * N * 0.001 * 17 = 1.017MN$ operations, where p is the hard location activation probability. Our model need more computing than the integer SDM in [3]. However, our operation is much simpler because hamming distance and simple addition operations are used in our model and Euclidean metric are used in integer SDM in [3].

5 Conclusion

This paper proposes a novel idea to extend the original SDM model to accept integer vectors instead of the binary vector using a neural coding scheme. It keeps the simplicity of the SDM implementation, but extends it to store integer vector patterns without introducing extra storage cost compared to integer SDM. Experimental evaluation shows that it can achieve satisfied performance on data retrieval and pattern recognition. It is a significant advantage that can be used for wide applications where integer feature patterns are used.

References

1. P. Kanerva, *Sparse distributed memory*. MIT press, 1988.
2. M. Mendes, M. M. Crisóstomo, and A. P. Coimbra, "Encoding data to use with a sparse distributed memory," in *Electronic Engineering and Computing Technology*, pp. 285–295, Springer, 2010.
3. J. Snaider and S. Franklin, "Integer sparse distributed memory," in *FLAIRS Conference*, 2012.
4. L. A. Jaeckel, "An alternative design for a sparse distributed memory." Report RIACS TR 89.28, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.
5. L. A. Jaeckel, "A class of designs for a sparse distributed memory." Report RIACS TR 89.30, Research Institute for Advanced Computer Science, NASA Ames Research Center, 1989.
6. T. A. Hely, D. J. Willshaw, and G. M. Hayes, "A new approach to kanerva's sparse distributed memory," *IEEE transactions on neural networks*, vol. 8 3, pp. 791–4, 1997.
7. U. Ramamaurthy, S. K. D'Mello, and S. Franklin, "Modified sparse distributed memory as transient episodic memory for cognitive software agents," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 6, pp. 5858–5863, IEEE, 2004.
8. H. Meng, K. Appiah, A. Hunter, S. Yue, M. Hobden, N. Priestley, P. Hobden, and C. Pettit, "A modified sparse distributed memory model for extracting clean patterns from noisy inputs," *2009 International Joint Conference on Neural Networks*, pp. 2084–2089, 2009.
9. P.-C. Huang and J. M. Rabaey, "A neuro-inspired spike pattern classifier," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, pp. 555–565, 2018.
10. F. Rieke and D. Warland, *Spikes: Exploring the Neural Code*. Bradford book, MIT Press, 1999.
11. M. S. Brogliato, D. M. Chada, and A. Linhares, "Sparse distributed memory: understanding the speed and robustness of expert memory," *Frontiers in human neuroscience*, vol. 8, p. 222, 2014.
12. D. Dua and C. Graff, "UCI machine learning repository." http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences, 2017.
13. C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, pp. 1–27, 2011.