Neural Networks 46 (2013) 144–153

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Integer sparse distributed memory: Analysis and results

Javier Snaider [a,*], Stan Franklin [a], Steve Strain [b], E. Olusegun George [c]

[a] Computer Science Department & Institute for Intelligent Systems, The University of Memphis, 365 Innovation Dr., Memphis, TN 38152, United States
[b] Department of Biomedical Engineering, The University of Memphis, 365 Innovation Dr., Memphis, TN 38152, United States
[c] Department of Mathematical Sciences, The University of Memphis, 365 Innovation Dr., Memphis, TN 38152, United States

ARTICLE INFO

Article history:
Received 3 July 2012
Received in revised form 5 May 2013
Accepted 6 May 2013

Keywords:
Sparse distributed memory
High dimensional space
Auto-associative memory

ABSTRACT

Sparse distributed memory is an auto-associative memory system that stores high dimensional Boolean vectors. Here we present an extension of the original SDM, the Integer SDM that uses modular arithmetic integer vectors rather than binary vectors. This extension preserves many of the desirable properties of the original SDM: auto-associativity, content addressability, distributed storage, and robustness over noisy inputs. In addition, it improves the representation capabilities of the memory and is more robust over normalization. It can also be extended to support forgetting and reliable sequence storage. We performed several simulations that test the noise robustness property and capacity of the memory. Theoretical analyses of the memory's fidelity and capacity are also presented.

1. Introduction

Sparse distributed memory (SDM) (Kanerva, 1988, 1993) is based on large binary vectors and has several desirable properties. It is distributed, auto-associative, content addressable, and noise robust. Moreover, this memory system exhibits interesting psychological characteristics as well (interference, knowing when it does not know, the tip of the tongue effect), that make it an attractive option with which to model episodic memory (Baddeley, Conway, & Aggleton, 2001; Franklin, Baars, Ramamurthy, & Ventura, 2005). Implementations of SDM are ongoing for various applications (Bose, Furber, & Shapiro, 2005; Furber, Bainbridge, Cumpstey, & Temple, 2004; Jockel, 2009; Mendes, Crisóstomo & Coimbra, 2009; Meng et al., 2009). Several improvements and variations have been proposed for SDM; for example Ramamurthy and colleagues introduced forgetting as part of an unsupervised learning mechanism (Ramamurthy, Baars, D'Mello, & Franklin, 2006; Ramamurthy & Franklin, 2011). The same authors also proposed the use of ternary vectors, introducing a "do not care" symbol as a third possible value for the dimensions of the vectors (D'Mello, Ramamurthy, & Franklin, 2005). Also Jaeckel (1989a, 1989b) proposed two variations of the original SDM, the selected coordinate design and the hyperplane design. Both designs modify the way that hard locations (see the next section) are selected. These designs slightly improve the signal to noise ratio of the memory. Furber and colleagues (2004) created a combined version of the Jaeckel's hyperplane design and a correlation matrix memory using sparkling neurons.

The original SDM uses binary vectors for both addresses and data words. This usage results in several limitations. First, real data are not always Boolean, making representations using more than two values desirable. A possible solution for this limitation is to use several dimensions of the word vectors to represent one feature, but this approach does not fit very well with the structure of SDM. In the distance calculation, a difference in any dimension has the same weight as that of any other dimension, but if several bits (i.e. dimensions) are used to represent a single feature, the weight of the bits should not be the same. Mendes and colleagues (2009) studied this problem in detail. They evaluated several binary encodings to use with SDM in robot navigation tasks and reported their difficulties and limitations. Using binary numbers coding some transitions have Hamming distances that incorrectly reflect the difference between the features. For example, the Hamming distance between seven (0111) and eight (1000) is 4 instead of the desired distance of 1. They also reported the performance of the Gray code (Gray, 1953), which only partially mitigates this effect. The best solution that they proposed is to use what the authors call *sum code*, in which, for example, 3 is represented as 111 and 5 as 11111. This coding substantially increases the dimensionality of the memory. Interestingly, they report that grouping bits and processing them as integers produces excellent performance. The extension proposed in this paper directly uses integer vectors, achieving similar performance but without the disadvantages reported by Mendes. While both models have essentially the same memory requirements, the vectors required by Mendes' model are larger (in bits) than the ones used by Integer SDM. Also, the dimensions in Mendes' are not homogeneous. Some dimensions have more "weight" than others, but this is not reflected in the metric employed in the space. Although we have

* Corresponding author.
E-mail address: jsnaider@memphis.edu (J. Snaider).

0893-6080/$ – see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.neunet.2013.05.005

not compared the experiments' results, the similarities between the two models suggest that they have similar capacity. However, the more homogeneous space and distance calculation of Integer SDM suggest that it would be more noise robust than Mendes' model.

Another disadvantage of binary vectors is the loss of information due to the noise introduced into the representation by the normalization used in combining vectors. Vectors can be summed up dimension by dimension (for this operation, vectors belonging to $\{0; 1\}^n$ are replaced by vectors of $\{-1; +1\}^n$). This operation produces a vector belonging to $\mathbb{Z}^n$, the space of integer vectors with n dimensions. The normalization process, which in general is a simple threshold function with a threshold of zero, reduces the resultant to a vector that is also in $\{-1; +1\}^n$ but with significant loss of information. See for example Kanerva (2009), Snaider and Franklin (2011, 2012a).

A variation of SDM, Integer Sparse Distributed Memory (Integer SDM), is based on large vectors, on the order of thousands of dimensions, where each dimension has a range of possible integer values. This memory has properties similar to the original one, noise robustness, auto-associativity and being distributed. A further extension of Integer SDM permits words and addresses of different lengths, which is particularly useful for the reliable storage of sequences and other data structures (Snaider & Franklin, 2011, 2012a). In addition, this memory avoids the limitations imposed by binary representation, as described above, allowing a better encoding of non-binary data and alleviating the normalization problem when combining several vectors.

The counters employed by the hard locations (see below) in the Integer SDM are similar, but not identical, to the hypercolumns described by Johansson and colleagues (2002). Each hypercolumn also has a predefined number of values, and the output follows winner-take-all rule, as in each dimension's output from a hard location. However, the learning mechanisms of the two models are different: the neural network with hypercolumns employs a mechanism similar to Hopfield networks, whereas Integer SDM's learning mechanism resembles that of the original SDM. Kanerva (1993) extensively compared these two models. Interestingly, Lansner and Holst (1996) described the use of hypercolumns to represent ranges of continuous values, which may also be represented using Integer SDM. Furthermore, they discuss the similarity between hypercolumns and the columns in the cortex, lending additional support to the biological plausibility of our model.

This paper is a follow up to Snaider and Franklin (2012b). In addition to the description of the memory, here we present detailed experiments that test the memory capacity and its noise robustness capability, and theoretical analyses of the memory fidelity and capacity, which constitute the main contributions of this paper. In the following section we briefly describe SDM. Then we introduce Integer SDM, present analyses of the memory fidelity and capacity, and discuss several experiments with this memory and their results. Finally we propose some directions for further research.

2. Sparse distributed memory

In this section, we briefly describe the components of SDM that are similar to those used in Integer SDM. For more information about SDM, both leisurely descriptions (Franklin, 1995, pp. 329–344) and highly detailed descriptions (Kanerva, 1988) are available.

SDM implements a content addressable random access memory with an address space of the order of 2^{1000} or even more. In the examples that follow, we will use 1000 dimensional binary vectors. Both addresses and words are binary vectors whose length equals the number of dimensions of the space. An important property of such high dimensional spaces is that two randomly chosen vectors are relatively far away from each other, meaning that they are uncorrelated. This vector space utilizes the Hamming distance to measure the distances between vectors. To construct the memory, a sparse uniformly distributed sample of addresses, on the order of 2^{20} of them, is chosen. The number of addresses selected to construct the memory is denoted by m. These addresses, called hard locations, are the units of storage of the memory. Only hard locations can store data. For this purpose, each hard location has counters, one for each dimension. To write a word vector in a hard location, for each dimension, if the bit of this dimension in the word is 1, the corresponding counter is incremented. If it is 0, the counter is decremented. To read a word vector from a hard location, we compute a vector such that, for each dimension, if the corresponding counter in the hard location is positive, 1 is assigned to this dimension in the vector being read; otherwise 0 is assigned.

A hard location can store several words, but as a combination rather than distinct entities. The reconstruction of one of these words requires the participation of many hard locations in its storage and retrieval. To read from an arbitrary address in SDM, the output vector is a composite of the readings of several hard locations. To determine which hard locations are used to read or write, an access sphere is defined. The access sphere for an address vector is a sphere with center at this address, enclosing, on average, a proportion p of the memory's hard locations; in our example 0.1% is used. To write a word vector in any address of the memory, the word is written to all hard locations inside the access sphere of the address. To read from any address, all hard locations in the access sphere of the address vector are read, and a majority rule for each dimension is applied.

In general, the SDM is used as an auto-associative memory, so the address vector is the same as the word vector (but see Snaider & Franklin, 2012a). In this case, after writing a word into the memory, the vector can be retrieved using partial or noisy data. If the partial vector is inside a critical distance from the original one, and it is used as an address with which to cue the memory, the output vector will be close to the original one. This critical distance depends on the number of vectors already stored in the memory. If the retrieval process is repeated, using the first recovered vector as address, the new reading will be even closer to the original. After a few iterations, typically fewer than ten, the readings will converge to the original vector. If the partial or noisy vector is farther than the critical distance away from the original one, the successive readings from the iterations will rapidly diverge. If the partial vector is about at the critical distance from the original one, successive iterations will yield vectors that are typically at the same critical distance from the original vector. As a result, the iterations circle the original vector, neither converging nor diverging. This behavior mimics the "tip of the tongue" effect.

Several authors have studied the capacity of SDM: Chou (1989), Kanerva (1988, 1993) and Keeler (1988). In particular Keeler compared the capacity of SDM to the capacity of a binary Hopfield net. He showed that both memories have the same capacity per storage element or counter. However, SDM presents an interesting advantage over Hopfield nets. In the former, the size of the words is independent of the number of storage elements; on the other hand, in the Hopfield nets the size of the words determines the capacity of the memory. Doubling the hard locations in SDM doubles the capacity of the memory for a given word size (Kanerva, 1993).

Willshaw networks (Willshaw, 1981) can achieve an information capacity of 0.69, which is higher than that for SDM. Knoblauch and colleagues (Knoblauch, Palm, & Sommer, 2010) extensively analyzed the performance of Willshaw networks and pointed out the importance of relating the capacity of associative memories with their fidelity (the probability of retrieving a written word). Comparing SDM to Willshaw networks in these terms can be interesting. However it is outside of the scope of this work.

SDM can be viewed as a synchronous, fully connected, three-layer, feed-forward artificial neural network (Kanerva, 1993). The