I J C T A, 10(5) 2017, pp. 939-945

A Sparse Distributed Neuro Fuzzy Architecture for Modeling and Analysis of Higher Order Systems

Anthony Doodnath* **and Musti Sastry***

Abstract: Higher order systems are complex to model, analyze and to control. This paper presents a new "Sparse Distributed Neuro Fuzzy (SDNF) Architecture" for higher order system modeling and analysis. This model consists of a few neuro fuzzy locations distributed within a sparse, high dimensional array. Addressing these neuro fuzzy locations are achieved with long vectors, since the dimensionality of the space is high, making it ideal for high order system modeling with some tolerance for errors in the long vector addressing. Even with high dimensionality and high number of neuro fuzzy locations within the space, it is demonstrated that realizing an output from the architecture only requires a few neuro fuzzy locations within a specific area of interest for precise determination, which saves time. Two different higher order dynamic systms were considered to evaluate the suggested SDNF architecture and results are presented.

Keywords: Neuro-fuzzy; tensor; sparse; high order; area of interest; SDNF.

1. INTRODUCTION

Higher order dynamic systems are complex, yet they exist in several areas that are commonly used. For example, dynamically changing values of one currency with respect to the other; ocean wave movements and oscillations – are few examples. Properties of higher order systems include – several variables, states, known and unknown boundary conditions, constraints etc. Modeling and analysis of higher order systems are complicated tasks. Numerical models have been used for higher order system computations. However, numerical methods are only good for estimation of the current or any specific state, but not for system modeling. Given the complexities and difficulties in higher order system modeling and analysis; several approaches have been attempted and reported in the literature thus far. John Maidens et. al proposed use of small information zones termed as 'kernel', which were identified using various langrangian computation techniques based on requirements of the system. Inaccuracy issues associated with higher order system modeling were also discussed [2]. These contributions included, but not limited to Particle Filter, lagrangian methods, stochastic methods etc. Brown et. al in 1995 outlined problems with high dimensional systems particularly in terms of higher degrees of dimensionality [5] and suggested a few approaches based on statistical models, partitioning techniques etc. However, most methods employ several approximations, conditions and assumptions due to which accuracy of prediction suffers[5,6]. One of the biggest problems is to model the higher dimensional systems with several variables in various dimensions, with an appropriate architecture [6, 7, 8]. Particle filter based system identification and tracking were suggested, however with a great degree of approximation in system identification [13 – 16] Use of tensors for multi-dimensional information systems is well-known. However, the applicability of tensors with a proper architecture for higher order dynamic systems is not very well studied. This paper addresses this gap. An attempt is made to combine tensors and neuro-fuzzy computational models to produce a new architecture to model and simulate higher order systems. The sparse distributed neuro fuzzy (SDNF) architecture can be described as a memory efficient storage and retrieval machine. It is inspired by a model called Sparse Distributed Memory, developed by Pentti Kanerva [1]. This was a binary memory model in pursuit of developing a

* Department of Electrical and Computer Engineering, The University of the West Indies, St. Augustine, Trinidad and Tobago

mathematical model of human long term memory. The SDNF architecture presented in this paper however uses real numbers, instead of binary numbers. At the onset, there are two properties of the SDNF architecture which allow it to function effectively:

1. Information stored within the SDNF architecture can be addressed through high dimensional or long vectors.
2. Any area within the space that information is stored is relatively far from any other areas of information within the space, i.e. the space is sparse.

In some ways, the SDNF Architecture acts as a random access memory in experiments conducted by Kong [3, 4]. The location of non-zero information within the space is called its address. The number of dimensions of the space N determines the length of the long vector which is used to address a location in the memory. For example a 50 dimensional space would require a long vector with 50 rows to address a location within the space. Since an area of interest within the space is relatively far from any other areas of interest, this enables some fault tolerance to long vectors, hence noisy or lost data would not necessarily result in an incorrect address to the space. The elements of the long vector are integers in the range 1-S, where S is the upper bound of each dimension of the space. For simplicity, each dimension is made same length so the space is a hypercube with each side having length S.

The space is sparse, and hence comprises mostly zeros. Sparse matrices are easily compressed resulting in significantly less computer data storage capacity than a dense matrix would utilize. The neuro fuzzy parameters are stored in the non-zero locations, which are called hard locations. These hard locations are distributed uniformly and randomly within the space.

Due to the sparseness of the space, using a random long vector address to access a hard location would practically never point to a hard location. Instead, when reading from location $\boldsymbol{x}$, all hard locations from within a certain distance a, from $\boldsymbol{x}$ are all activated. The output is the sum of all the individual outputs of the hard locations within distance a, of the location $\boldsymbol{x}$. While the distance a can take any value between 1-S, careful selection of the distance a can make the architecture fault tolerant for noisy or lost data in the long vector address since a determines the size of the area of interest, and each area of interest is relatively far away from any other areas of interest in the space.

A call to location $\boldsymbol{x}$ activates only few hard locations of the space. When the location $\boldsymbol{x}$ is called, only those few locations within distance a of $\boldsymbol{x}$ therefore are activated, and the remaining hard locations are not called. This is achieved by a special address decoder function which presents the long vector $\boldsymbol{x}$ to the address of all the hard locations in the space in parallel, but only those neurons which are within distance a of $\boldsymbol{x}$ fires. This parallel computation is the basis of the efficiency of the architecture, where the determination of hard locations within the area of interest does not require an exhaustive search of all hard locations within the space.

From the above, it can be seen that addressing a specific location in a higher order system involves data or information retrieval, processing, and related computation. Tensors can be effectively used to do this. Also, it can be seen that other attempts to address the problem with high dimensionality and rule explosion in neuro fuzzy systems have been proposed in [4, 5, 6, 7] which proposes rule pruning and order reduction that leads to a loss in data. The following sections will illustrate the proposed SDNF architecture and its application to higher order problems.

2. NEURO FUZZY MODEL

The SDNF architecture can be described using vector and matrix notation. The non adaptive (fixed) parameters used to initialize the SDNF architecture are given by:

N – Address length; number of dimensions of the address space; input dimension.

S – The length of each side of the hypercube space.

a – The threshold distance between hard locations within an area of interest to an address in the space.

H – The number of hard locations of the SDNF space.

Each hard location contains the adaptable neuro fuzzy (NF) parameters set $\{\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{p}\}$ where

$$c = [c_1, c_2, ..., c_N] \quad c \in \{1, 2, ..., S\} \tag{1}$$

$$b = [b_1, b_2, ..., b_N] \quad b \in \{R\} \tag{2}$$

$$p = [p_1, p_2, ..., p_{N+1}] \quad p \in \{R\} \tag{3}$$

Writing data to an address $\boldsymbol{x}$ involves writing the NF parameters $\{\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{p}\}$ to all the hard locations within the threshold distance *a* of $\boldsymbol{x}$.

Reading from an address $\boldsymbol{x}$ is the sum of the neuro fuzzy output of all the hard locations within the threshold distance *a* of $\boldsymbol{x}$, as illustrated in Figure 1.

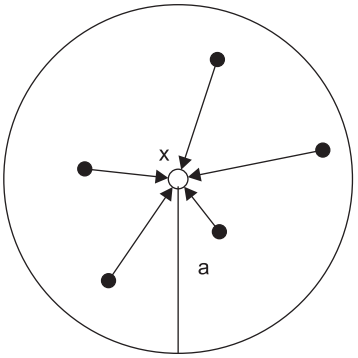

Figure 1: Hard locations within threshold distance constitute the output of reading from address *x*

The contributory fuzzy output in reading from an address $\boldsymbol{x}$, for each hard location [7] is realized by the following:

$$y = \sum_{j=1}^{N} w_j f_j \tag{4}$$

$$f_j = \sum_{j=1}^{N} [p_j (x_j - c_j)] + p_{N+1} \tag{5}$$

$$w_j = \prod_{j=1}^{N} \frac{1}{1 + \beta \left| \frac{x_j - c_j}{a} \right|^{2bj}} \tag{6}$$

If there are *m* hard locations in the area of interest within distance *a* of $\boldsymbol{x}$, then the total output is given by:

$$y_T = \sum_{i=1}^{m} y_i \tag{7}$$

3. STORAGE AND RETRIEVAL OF NEURO FUZZY PARAMETERS

The SDNF architecture is initialized with randomly distributed hard locations in a vastly sparse array. The parameters stored in these locations are $\{\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{p}\}$, where c is the address of the hard location. Parameters b and p are the neuro fuzzy parameters used to calculate the neuro fuzzy output from equations 5 and 6.

The threshold distance a is key to the efficiency of read and write operations. As the distance a increases from 1 to S, a small sub-hypercube with the same dimensionality of the SDNF space grows from a single element (a = 1) to encompass the entire space (a = S). The capacity of the sub-hypercube with respect to the SDNF capacity is initially infinitesimal, until the distance reaches some critical value, where it can be likened to a "bubble" that suddenly pops into existence such that the area of interest becomes discernable in comparison to the overall space as illustrated in Figure 2. Below this critical distance, the probability that there are hard locations within the area of interest is almost zero and the SDNF output would be equal to zero. In other words, Figure 2 provides the relationship between the threshold distance a and its ability to be discerned as a significant sub-entity of a 10-dimensional SDNF space with each dimension 50 units long.

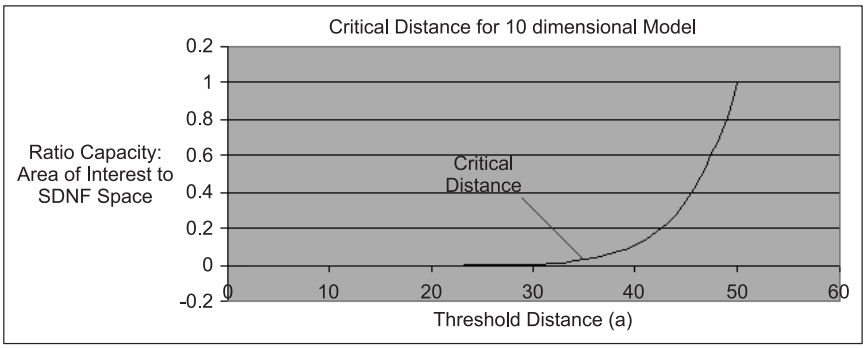

Figure 2: Relationship between the threshold distance and Capacity ratio

If a is too close to S, then the number of hard locations within the area of interest m would be close to H, the total number of hard locations in the space. Computing the output would be computationally exhausting, similar to realizing an output from an artificial neural network where all the neurons are evaluated for all inputs. For efficiency, the threshold distance is therefore chosen just above the critical distance such that no more than a few hard locations are to be evaluated for any input.

Once the critical distance a is selected, this is set in the address decoder function. This function presents the input to the addresses of all the hard locations in parallel as shown in Figure 3. Only the hard locations with addresses within the threshold distance of the long vector input fires.

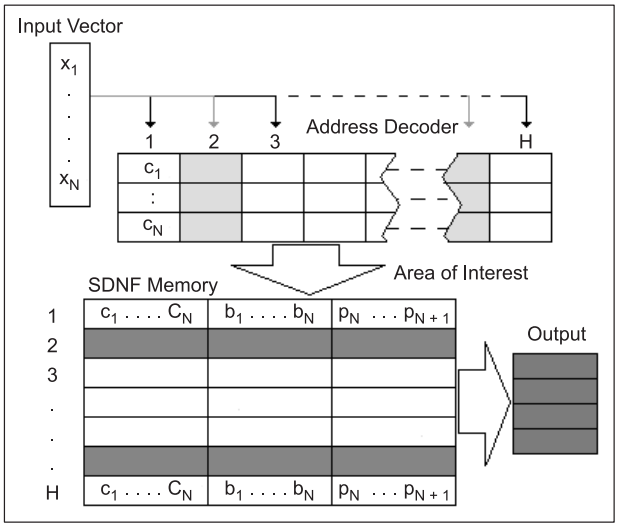

Figure 3: Instant addressing of locations within area of interest from the SDNF Memory

When the SDNF is initialized, the address of the hard locations $\boldsymbol{c}$, are pre-allocated as well as the $\boldsymbol{b}$ and $\boldsymbol{p}$ fuzzy parameters. Writing data to an address is done by presenting an *[address output]* pair to the SDNF model. An error function between the SDNF output and the write output is then iteratively back propagated to the SDNF model, where the locations $\{\boldsymbol{c}, \boldsymbol{b}, \boldsymbol{p}\}$ parameters of the hard locations within the area of interest are updated until the error is minimized. The data is therefore distributed among the several hard locations within the area of interest.

4. TESTING AND RESULTS

The SDNF model was applied to model two standard, higher order complex models, which are well-known time-series based models. These are Mackey Glass time series [17, 18] and US Dollar to British Pound Exchange Rate [16]. Both of these models are considered as a good benchmarking systems for both high dimensional and chaotic systems The parameters for the test were as follows: S = 100, N = 5, a = 30 (critical distance) and H = 200. The SDNF model was used to predict the next value in the time series given a sequence of its previous outputs, so SDNF input was previous values in the time series and the SDNF output was the next value in the time series.

Mackey Glass time series model is considered for the first case study. This percentage root mean square (RMS) error between the SDNF output and the Expected output was 5%. Figure 4 shows the simulated response for the Mackey Glass time series, where the SDNF output closely tracks the Expected Output.

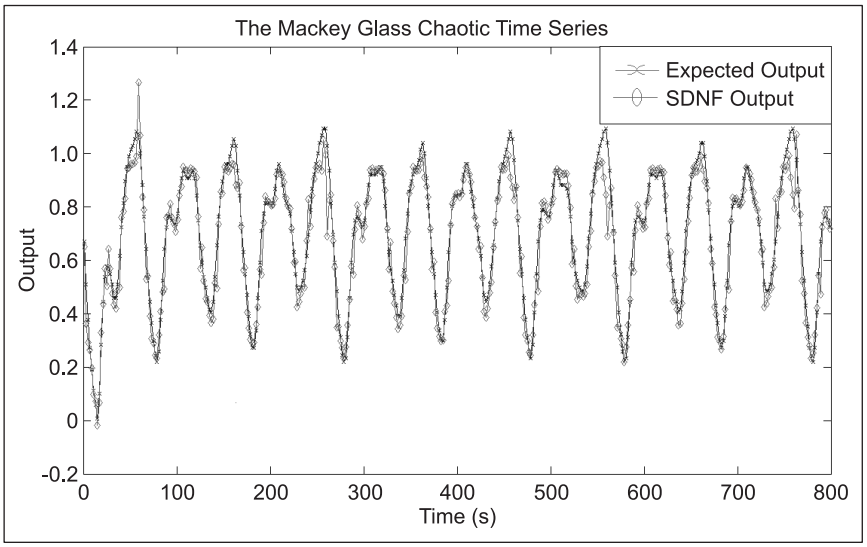

Figure 4: The Response of the SDNF Model to the Mackey Glass Time Series

US Dollar to British Pound Exchange Rate data series model specifically during the period 1981 to 2005 is considered for the second case study. The proposed SDNF was applied to this time series model to test the adaptability of the SDNF model is real data from the

The percentage RMS error between the SDNF output and the Expected output was 6%, and the output response is shown in Figure 5 below. This graph shows close tracking between the SDNF output and the expected output.

5. DISCUSSION

Evaluation of the sparse array demonstrated that an average of about 5-10 hard locations are called for each read operation from the SDNF model, so not all 200 hard locations are required to be evaluated for each read operation.

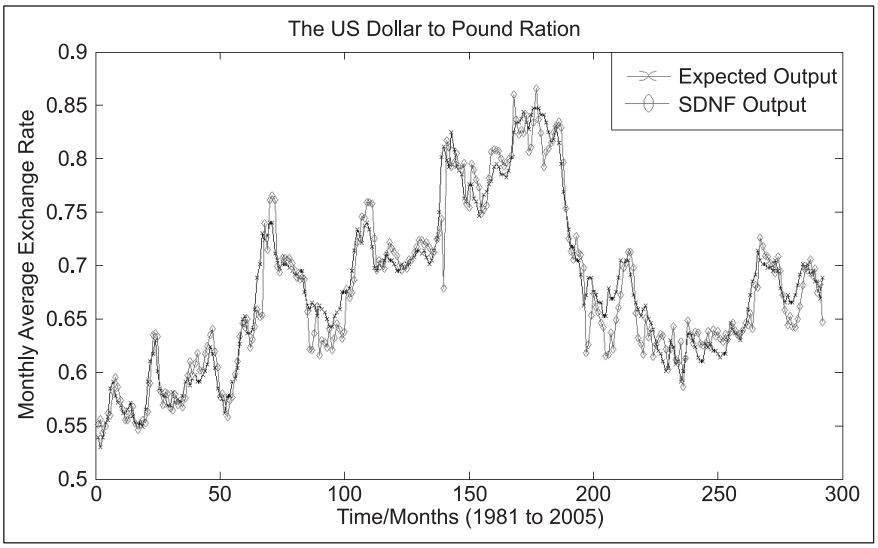

Figure 5: The Response of the SDNF Model to real data for the Exchange Rate between the US Dollar and the British Pound

The accuracy of the output can be increased by increasing the number of hard locations within the SDNF model, thereby increasing the average number of hard locations within each area of interest.

The SDNF model has proven to be able to model complex nonlinear processes within 5% accuracy, as shown by the responses of Figures 4 and 5. Both models are chaotic, non-linear time series, which are difficult to model using traditional neuro fuzzy techniques.

As the number of dimensions N is increased, the critical distance increases, and hence the threshold distance has to be increased in order to determine an output.

Higher dimensional models have proven save on CPU memory, both RAM and ROM. This is a result of the way in which sparse matrices are stored where pre-allocation of memory is done for only non-zero locations.

The address decoder function which was developed has been developed based on neural networks working in parallel, where the same input is presented to the same network layer at the same time. This has proven more computationally efficient than sequentially calculating the distance of each hard location to the area of interest.

The way in which the address decoder operates resembles the response of the human brain to stimuli. For example the address decoder tolerates small errors in the input vector, similar to the way in which a visual object known by the brain can be recognized even though it is hardly likely that it is seen from the same perceptive. That is, the input to the brain is tolerant to small variances to its input.

The distributed nature of the data that is stored in the SDNF model has its origin from observations that certain brain cells may die and/or regenerate but the data is not lost because the knowledge is thought to be distributed among a cluster of cells. This approach is computationally efficient than traditional neuro fuzzy techniques which require exhaustive realization of all the neuro fuzzy elements to achieve an output.

The SDNF model therefore is thought to be a suitable container for mapping of human knowledge in trying to develop a model for the human brain, as well as other high order models which have previously thought to be elusive to the field of neuro fuzzy techniques.

6. CONCLUSION

The need and basis for a sparse distributed neuro fuzzy model which is applicable for high dimensional system modeling has been proposed. Its conceptualization and construction has been timely since the curse of dimensionality and explosion of rules have formed a virtual glass ceiling in applications of high dimensional neuro fuzzy techniques. The architecture has been demonstrated to be computationally efficient, and its application to model two well-known complex nonlinear processes has been presented.

References

1. Kanerva, Pentti. "Sparse Distributed Memory," Cambridge Massachusetts, The MIT Press, 1988.
2. John N. Maidens et. al, "Lagrangian methods for approximating the viability kernel in high-dimensional systems", 2013, Automatica, Vol. 49, pp. 2017-2029
3. A. Kong, "Impulse Activated Sparse Cell Array Network in Non-linear Autoregressive Process Modeling," Soft Computing - A Fusion of Foundations, Methodologies and Applications: Springer, 2005, pp 421-429.
4. A. Kong, "Sparse Distributed Fuzzy Inference Systems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*. Springer, 2006, pp 567-577.
5. M. Brown, K.M. Bossley, D.J. Mills and C.J. Harris, "High Dimensional Neurofuzzy Systems: Overcoming the Curse of Dimensionality" Fuzzy Systems, International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Yokahama Japan, 1995, pp 2139-2146.
6. D.J. Mills, K.M. Bossley, M. Brown, M. and Harris, C. J., "Towards Parsimonious High-Dimensional Neurofuzzy Systems," World Congress on Neural Networks 1995, pp 717-720.
7. J. Weinschenk, W.E. Combs, R.J. Marks, "Avoidance of rule explosion by mapping fuzzy systems to a union rule configuration'" The 12th *IEEE International Conference on Fuzzy Systems*, Volume 1, 2003, pp 43-48.
8. J.S.R. Jang, C.T. Sun, and E. Mitzuni, "Neuro-Fuzzy and Soft Computing – A computational Approach to Learning Machine Intelligence", Prentice Hall Inc., NewJersey, 1997.
9. Bader, Brett W., and Tamara G. Kolda. 2007. Efficient MATLAB computations with sparse and factored tensors. SIAM Journal on Scientific Computing 30 (1): 205-231.
10. Baskaran, Muthu, Benoit Meister, Nicolas Vasilache, and Richard Lethin. 2012. Efficient and scalable computations with sparse tensors. In *IEEE conference on high performance extreme computing* 10-12 September, 2012, 1-6. New Jersey, Doi 10.1109/HPEC.2012.6408676.
11. Doodnath, Anthony, Albert Kong, and Musti Sastry. 2009. Optimal Control of Blood Glucose. In Proceedings of the 2009 I*EEE WRI world conference of computer science and information engineering*, March 31 – April 2, 2009, 337-381. Los Angeles, Doi: 10.1109/CSIE.2009.380.
12. Jonathan Poterjoy, "A Localized Particle Filter for High-Dimensional Nonlinear Systems", 2016, American Meteorological Society, Vol. 144, pp. 59-76, DOI: 10.1175/MWR-D-15-0163.1.
13. Petar M. Djuric, M ´ onica F. Bugallo, "Particle Filtering for High-Dimensional Systems", 2013, 5th *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 352-355.
14. Jing-Ling Chen et. al, "Hardy's paradox for high-dimensional systems", PHYSICAL REVIEW A, Vol. 88, No. 6, 2013, pp. 1-5.
15. M. Ades, and P. J. van Leeuwen, "The equivalent-weights particle filter in a high-dimensional system", *Quarterly Journal of the Royal Meteorological Society*, Vol. 141, pp.484-503, 2015.
16. H. He and X. Shen, "Bootstrap Methods for Foreign Currency Exchange Rates Prediction," 2007 *International Joint Conference on Neural Networks*, Orlando, FL, 2007, pp. 1272-1277. doi: 10.1109/IJCNN.2007.4371141.
17. P. Amil, C. Cabeza and A. C. Marti, "Exact Discrete-Time Implementation of the Mackey–Glass Delayed Model," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, Vol. 62, No. 7, pp. 681-685, July 2015. doi: 10.1109/TCSII.2015.2415651.
18. Dongping Wang and Juebang Yu, "Chaos in the fractional order Mackey-Glass system," 2008 *International Conference on Communications, Circuits and Systems*, Fujian, 2008, pp. 641-645. doi: 10.1109/ICCCAS.2008.4657855.