Proceedings of the 4th WSEAS Int. Conf. on Information Security, Communications and Computers, Tenerife, Spain, December 16-18, 2005 (pp466-471)

Non-Linear Data for Neural Networks Training and Testing

ABDEL LATIF ABU-DALHOUM MOHAMMED SADIQ AL-RAWI
Computer Science Dept,
King Abdullah the Second School for Information Technology.
Jordan University
Amman 11942, P.O. Box 13496, JORDAN

Abstract: - Highly nonlinear data sets are important in the field of artificial neural networks. It is not feasible to design a neural network and try to classify some real world data directly with that network. N-bit parity is one of the oldest data used to train and test neural networks. The simplest is the 2-bit parity also known as the XOR classification problem. Some researchers say that N-bit parity s set though highly nonlinear it is a simple task to learn by neural networks, others were drifted to tailor special purpose neural networks to solve only the N-bit parity problem without explaining why there is such a need. Is it possible to judge the N-bit parity is a simple data due to the fact that it can be modeled by a deterministic finite accepter? Moreover, should patterns that are in the form of context free which require a pushdown automaton, or context-sensitive and recursively enumerable that require a Turing machine be harder to learn by neural networks? The aim of this paper is to focus on and propose some complex nonlinear data to be used in training and testing of neural networks. The most important in these parity data is that the developer can tune the complexity of nonlinearity through various amounts of degrees; the user can select various numbers of categories, huge number of pattern samples, and many hybrid symbols. Testing for various neural networks and their generalization and ability to classify unseen patterns can be more effective. Experimental results on the classification of prime numbers showed that neural networks can learn the classification of prime numbers.

Key-Words: - Neural Networks, nonlinear separable, hybrid N-parity, prime numbers classification, E1 problem, All symbol-parity, hybrid symbol-parity.

1 Introduction

The N-bit parity is a typical classification problem addressed in neural networks literature. N-bit parity is a mapping defined on 2^N distinct binary vectors that indicates whether the sum of the N components of a binary vector is odd or even (if the sum is odd the input vector is classified as the first category, else input vector belongs to the second category). This simply constructed data, linearly inseparable, was used by various researchers in training and testing neural networks.

The parity problem is a very difficult task for neural networks to learn with generalization [1][2][3][4]. The aim of N-bit parity is to test neural network approaches that would be able to solve problems of unknown linearity. Neural networks should do this automatically without any elegant network topology design. What is the importance of carefully designing a network to solve the N-bit parity, especially if such networks cannot be trained with backpropagation (or any other training method). One question introduced in this literature whether N-bit parity classification has any direct real world application? No answer can be found, and if so, we can perform the task with logical operators, or a simple counter. The simplest parity problem is the 2-bit parity (also known as the XOR problem). The classical Parallel Distributed Processing textbook [1] states that a multi layer feedforward neural network trained with backpropagation needs at least N hidden layer neurons to solve the N-bit parity. Recent works show that using some shortcut connections and/or non-standard activation function(s) the N-bit parity could be solved with less than N hidden neurons [2]. In fact, the problem can be solved with one output neuron if a non-differentiable activation function is used [3]. More sophisticated studies [3][4] used the triple parity such that three symbols are used {0,1,2} to generate strings, and if 0's, 1's, and 2's are even they are classified (accepted) as the target class, otherwise they should be rejected. The triple parity benchmark had been used to test generalization and pruning of

recurrent neural networks. Many works [5] emphasize that N-bit parity is a simple task for the neural network to learn and can be performed via a random search method on a recurrent neural network. Therefore, despite the complexity of N-bit parity, they [5] argued that this is a simple problem for neural networks to learn and more complicated (nonlinear) data are needed to train and test neural networks. However, [6][7][8][9][10][11][12][13] showed that many neural networks experimentations have been performed using N-bit parity data.

The aim of this paper is to propose many data sets with complex nonlinearity. Some data may be derived from regular, context-free, and/or context sensitive languages (knowing that N-bit parity is a form of regular language). All the proposed data sets are highly nonlinear and they provide a large data benchmark for training and testing neural networks.

2 Data with Complex Nonlinearly

To investigate neural networks ability to learn with generalization (their ability to classify unseen samples), researchers used N-bit parity and other artificial data. In this section, we propose several other data sets based on N-symbol-parity (N-parity) rather N-bit parity. The N-parity is more complicated than N-bit parity due to the use of many symbols in the parity check. Some of the proposed N-Parity problems are as described in the following sections.

2.1 Hybrid N-Parity problem (HNP)

This is a data set that mixes symbols that are used for parity check {-1, 1} and the one that is not used in parity check {0} the HNP. This set is defined as follows

Definition of **HNP**: Let $\sum = \{-1, 0, 1\}$ be the set of alphabets used to derive the string $\mathbf{s}$ such that $\mathbf{s} \in \sum^+$, $|\mathbf{s}| = N$. If the total number of categories is four $\omega_1, \ldots, \omega_4$ that denotes the first to the fourth category respectively, and -1's and 1's are counted for parity then the four categories are as given below

$$\begin{aligned} \mathbf{s} \in \omega_1 &: \quad n_1(\mathbf{s}) \text{ is odd and } n_{-1}(\mathbf{s}) \text{ is odd} \\ \mathbf{s} \in \omega_2 &: \quad n_1(\mathbf{s}) \text{ is odd and } n_{-1}(\mathbf{s}) \text{ is even} \\ \mathbf{s} \in \omega_4 &: \quad n_1(\mathbf{s}) \text{ is even and } n_{-1}(\mathbf{s}) \text{ is odd} \\ \mathbf{s} \in \omega_3 &: \quad n_1(\mathbf{s}) \text{ is even and } n_{-1}(\mathbf{s}) \text{ is even} \end{aligned}, \qquad (1)$$

where $n_1(\mathbf{s})$ and $n_{-1}(\mathbf{s})$ represent the number of 1's and -1's in $\mathbf{s}$ respectively

Note: $\sum^+ = \sum^* - \varepsilon$, where ε is an empty string and $\sum^*$ denotes the set of strings (pattern samples in this paper) obtained by concatenating ε or more symbols from $\sum$. It is obvious that the total number of samples is n^N. Neural networks training and testing for the HNP have been discussed previously in [14].

2.2 All symbol parity (ASP)

This is a modification of the classical N-bit parity. The AS P is defined as follows:

Definition of **ASP**: Let $\sum = \{a_0, a_1, \ldots, a_{n-1}\}$ be the set of alphabets used to derive pattern samples such that a pattern sample $\mathbf{s} \in \sum^+$ with $|\mathbf{s}| = N$. If $n = |\sum|$, then the total number of pattern samples is n^N that are classified into c categories as follows

$$\begin{aligned} \mathbf{s} \in \omega_1 &: \quad D(\mathbf{s}) = d_1 \\ \mathbf{s} \in \omega_2 &: \quad D(\mathbf{s}) = d_2 \\ &\vdots \qquad\quad \vdots \\ \mathbf{s} \in \omega_c &: \quad D(\mathbf{s}) = d_c \end{aligned}, \qquad (2)$$

where $d_i \in \{0, 1, \ldots, 2^n - 1\}$ for $i = 1, \ldots, c$ such that $d_1 < d_2 < \ldots < d_c$, $D(\mathbf{s}) = \text{DecimalOf}(p)$, and $p = b_{a_{n-1}}(\mathbf{s}) b_{a_{n-2}}(\mathbf{s}) \ldots b_{a_1}(\mathbf{s}) b_{a_0}(\mathbf{s})$ is a binary number concatenated from $b_{a_t}(\mathbf{s})$ which is given by

$$b_{a_t}(\mathbf{s}) = \begin{cases} 0 & \text{the number of } a_t \text{ in } \mathbf{s} \text{ is odd} \\ 1 & \text{the number of } a_t \text{ in } \mathbf{s} \text{ is even} \end{cases}, \qquad (3)$$

for $t = 0, 1, 2, \ldots, n-1$. It is obvious that the above ASP offers a wide range of nonlinearity with different degrees of complexities. May be it is better to express the ASP as a function of N and n, therefore, we write $\text{ASP}(N, n)$. It is easy to show that $c = 2^n$ for $N = n$. Below, we demonstrate a simple example of generating some ASP data set. The data generated is highly nonlinear and other higher orders can be generated too.

Example 1: Let $\Sigma = \{0, 1, 2, 3, 4\}$, $N = |\mathbf{s}| = 2$, and $n = 5$ then the pattern samples $\{00, 01, 02, \ldots, 44\}$ of

ASP(2,5) are classified into eleven categories as shown below in Table 1:

Table 1: The ASP(2,5) data set

· 1	· 2	· 3	· 4	· 5	· 6	· 7	· 8	· 9	· 10	· 11
00, 11, 22, 33, 44	01, 10	02, 20	03, 30	04, 40	12, 21	13, 31	14, 41	23, 32	24, 42	34, 43

2.3 Hybrid symbol N-Parity (HSNP)

This is yet another modification to the ASP in which some of the symbols are only padded to the pattern samples without being counted for parity check, the same as the {0} role in the HNP. The definition of HSNP is as follows

Definition of **HSNP**: Let $\sum = \{a_0, a_1, \ldots, a_{n-1}\}$ be the set of alphabets used to derive pattern samples such that a pattern sample $\mathbf{s}. \Sigma^+$ with $|\mathbf{s}| = N$. The included set . to be used in the parity check is a subset from $\sum$ and can be selected as desired, let $. = \{a_i, a_j, ..., a_r, a_s\}$, such that $k = |. |$. The set of symbols excluded from parity check is $. = \sum -.$, and $|. | = n - k$. If $n = |\Sigma|$, the total number of pattern samples is n^N that are classified into c categories as follows:

$$\begin{aligned} &\mathbf{s}. .{}_1 : . \quad && D(\mathbf{s}) = d_1 \\ &\mathbf{s}. .{}_2 : . \quad && D(\mathbf{s}) = d_2 \\ &\quad \vdots && \quad \vdots \\ &\mathbf{s}. .{}_c : . \quad && D(\mathbf{s}) = d_c \end{aligned}, \qquad (4)$$

where d_i is a decimal integer obtained from $d_i . \{0,1,...,2^k - 1\}$, for $i = 1,...,c$ such that $d_1 . d_2 . \ldots . d_c$, $D(\mathbf{s}) = \text{DecimalOf}(p)$, and $p = b_{a_i}(\mathbf{s})\ b_{a_j}(\mathbf{s}) \ldots b_{a_r}(\mathbf{s})\ b_{a_s}(\mathbf{s})$ is a binary number concatenated from $b_{a_t}(\mathbf{s})$ which is given in (3). It is easy to show that $c = 2^k$ for $N = k$.

Yet another wide range of complex nonlinearity, the user should select how many symbols to include in the parity check. Again, it is better to superscript the HSNP to be $\text{HSNP}(N, n, k)$. To such a case, the HNP is defined as $\text{HSNP}(\text{N}, 3, 2)$, where $\sum = \{-1, 0, 1\}$, $. = \{-1, 1\}$, $. = \{0\}$, $|\mathbf{s}| = N$. Moreover, $\text{ASP}(\text{N,n}) = \text{HSNP}(N, n, n)$ that is including all the symbols results in the ASP. Also, HSNP(*N*, 2, 1) for $\sum = \{0,1\}$, $. = \{1\}$, $. = \{0\}$, $|\mathbf{s}| = N$ is the N-bit parity data set. Last but not least, HSNP(2,2,1) is the so called XOR data set. For bipolar data, we can replace $\sum = \{0,1\}$ with $\sum = \{-1,1\}$ where $. = \{-1\}$ without any problem. The below example demonstrates the generation of HSNP(2,5,2).

Example 2: Let $\Sigma = \{0,1,2,3,4\}$, $n = 5$, $. = \{1,2\}$, $k = 2$, and $N = |\mathbf{s}| = 2$, then the pattern samples $\{00, 01, 02, ..., 44\}$ are classified into four categories (since $N = k$ we will have 2^k categories which is four) according to HSNP(2,5,2) as shown below in Table 2:

Table 2: The HSNP (2,5,2) data set

· 1	· 2	· 3	· 4
00, 11, 22, 33, 44, 03, 30, 04, 40, 34, 43	01, 10, 14, 41, 13, 31	02, 20, 23, 32, 24, 42	12, 21

2.4 Dichotomized hybrid symbol N-Parity (DHSNP)

Another modification can be done to the $\text{HSNP}(N,n,k)$ such that the number of categories is two as follows:

Definition of **DHSNP** Let $\sum = \{a_0, a_1, \ldots, a_{n-1}\}$ be the set of alphabets used to derive pattern samples such that a pattern sample $\mathbf{s}. \Sigma^+$ with $|\mathbf{s}| = N$. The included set . to be used in the parity check is a subset from $\sum$ and can be selected as desired, let $. = \{a_i, a_j, ..., a_r, a_s\}$, such that $k = |. |$. The set of symbols excluded from parity check is $. = \sum -.$,

therefore, $|.| = n-k$. If $n = |\Sigma|$, then the total number of pattern samples is n^N that are classified into two categories as follows

$$\begin{array}{ll} \mathbf{s} \in \omega_1: & \mathrm{n}_1(p) \text{ is even} \\ \mathbf{s} \in \omega_2: & \mathrm{n}_1(p) \text{ is odd} \end{array}, \quad (5)$$

where $p = b_{a_i}(\mathbf{s})\ b_{a_j}(\mathbf{s}) \ldots b_{a_t}(\mathbf{s}) .. b_{a_r}(\mathbf{s})\ b_{a_s}(\mathbf{s})$ is a binary number concatenated from $b_{a_t}(\mathbf{s})$ which is given in (3).

2. 5 The =1 (E1) problem

The =1, is a famous data set used with neural networks. We can define the =1 problem according to the previous notations as below

Definition of E1: Let $\sum = \{a_0, a_1, \ldots, a_{n-1}\}$ be the set of alphabets used to derive pattern samples such that a pattern sample $\mathbf{s} \in \sum^+$ with $|\mathbf{s}| = N$. The included set is $\Omega = \{a_j\}$ where $a_j \in \Sigma$. The total number of pattern samples is n^N can be classified into two categories as follows

$$\begin{array}{ll} \mathbf{s} \in \omega_1: & \text{the number of } a_j \text{ in } \mathbf{s} \text{ is } 1 \\ \mathbf{s} \in \omega_2: & \text{otherwise} \end{array}. \quad (6)$$

The famous (=1) problem is the one such that $a_j = 1$ and all other alphabets are set to 0 which is a subset of the E1 problem given above.

2.6 Prime number problem (PNP)

This data set contains the binary representation of decimals. The task is that neural network should learn the classification a decimal numbers into prime numbers or not.

Definition of **PNP**: Let $\sum = \{0,1\}$ be the set of alphabets used to derive the string $\mathbf{s}$ such that $\mathbf{s} \in \sum^+$, if the total number of categories is two; ω_1 denotes the first category and ω_2 denotes the second category, then

$$\begin{array}{ll} \mathbf{s} \in \omega_1: & \text{DecimalOf}(\mathbf{s}) \text{ is prime} \\ \mathbf{s} \in \omega_2 & \text{otherwise} \end{array}. \quad (7)$$

A neural network that can learn to classify the above problem would be a marvels prime number generator (provided that it is efficient) which can be used in number theory and in applications related to public key encryption, or symmetric encryption. Below is an example of a limited set.

Example 3: Let $\Sigma = \{0,1\}$, $|\mathbf{s}| = 3$, then the PNP via binary representation as inputs to neural networks. Three inputs should be used for the neural network, see the below PNP data in Table 3.

Table 3: The PNP for n=2.

ω_1				ω_2			
000	100	001	011	110	010	101	111

Example 4: Using decimal representation (0 to 7) as inputs to neural networks, only one input is needed. See Table 4 for illustration.

Table 4: The prime classification problem using decimal representation

ω_1				ω_2			
0	1	4	6	2	3	5	7

The PNP will be used in neural network training and testing in this paper in order to show the complexity and the learning ability of this data.

3 Experimental Results

The PNP data set presented in this paper has been used in training and testing many neural networks with different networks topologies. The table below demonstrates the performance of using the PNP data set in neural networks training and testing. As for neural networks, we used a feedforward neural network [11] in training and testing. SCG 16-10-1 refers to the neural network such that the number of inputs is one, number of hidden neurons is 10, and, the number of outputs is one (the input string is either prime or not). The method used for optimizing the neural network is the conjugate gradient descent [12][13]. Results obtained using the PNP for strings of size 16 are shown in Table 5.

As shown in Table 5, decimal numbers are converted into binary each of length 16 bit, then strings representing decimals from 0 to 1000 have been used in training, and strings representing decimals from 3000 to 40000 and strings representing decimals from 64000 to 65000 have been used in testing. The aim is to classify a decimal as being prime number or not. In Table 6, we

implemented the same experiments and show that using decimal numbers as training data for neural networks, the network is unable to classify decimal numbers into prime or not. No successful learning happens and all networks we tried did not converge.

Table 5. Experimental results of the classification of decimal numbers into prime or non-prime according to their binary string representation (the PNP problem). SCG 16-10-1 stands for the feedforward neural network with 16 inputs, 80 hidden neurons, and one output neuron that is trained with scaled conjugate gradient.

Neural Network Used/ Topology	*Training set Decimal numbers from-to*	*Testing set Decimal numbers from-to*	*Correct Recognition on the testing set*	*Min Error goal*
SCG 16-20-1	0-1000	0-1000	983/1001	0.06
SCG 16-10-1	0-1000	0-1000	995/1001	0.13
SCG 16-10-1	0-1000	3000-4000	807/1001	0.13
SCG 16-10-1	0-1000	64000-65000	819/1001	0.13
SCG 16-80-1	0-1000	3000-4000	812/1001	0.03
SCG 16-80-1	0-1000	0-1000	993/1001	0.03
SCG 16-5-1	0-1000	64000-65000	730/1001	0.02
SCG (#0) 4-2-1	0-15	0-15	16/16	2E-7

(#0) In this experiment, only 4 bits are used since the decimals are from 0 to 15.

Table 6. Experimental results of the classification of decimal numbers into prime or non-prime according to decimal value representation. SCG 1-5-1 stands for the feedforward neural network with one input, five hidden neurons, and one output neuron that is trained with scaled conjugate gradient.

Neural Network Used/ Topology	*Training set Decimal numbers from-to*	*Testing set Decimal numbers from-to*	*Correct Recognition on the testing set*	*Min Error reached*
SCG 1-5-1	0-15	0-15	7/16	0.40
SCG 1-20-1	0-15	0-15	16/16	4E-7
SCG 1-80-1	0-1000	0-1000	18/1001	0.54 (#1)
SCG 1-160-1	0-1000	0-1000	733/1001	0.48
SCG (#2) 1-80-1	0-1000	0-1000	200/1001	0.52
SCG (#2) 1-160-1	0-1000	0-1000	413/1001	0.45

(#1) The minimum error after training is 0.5, no convergence happened within more than 20000 epochs. (#2) Statistical normalization to zero mean and unit variance has been performed so that all decimals are real numbers lying in between -1 and 1.

The prime number classification needs more work to improve the accuracy of the classification. As can be seen in Table 5, of accuracy classifying prime numbers is more than 95% for using the training set in testing, while obtaining more than 80% of accuracy when using disjoint training and testing sets.. Each of the above experiments is repeated up to 10 times and the recognition accuracy shown is the average. Thousands of samples have been used in training and testing.

Many other experiments have been performed using only decimals from 0 to 15. In this experiment the neural network did learn both the decimal representation data set and the binary (PNP) representation set. The failure for neural networks to learn large numbers that are presented as direct or normalized decimals suggests that the neural network (when learning small set of decimals) is working as an associative memory knowing that the number of weights of the network in this case is greater than the learned decimals.

4 Conclusions

For any classification problem, the computer should perform heuristic searches on neural networks to find the optimum (or best) weights and topology with little human intervention as much as possible. In addition to the proposed new data sets, we conclude that more elaborated work should be done to train and test neural networks, by tackling high nonlinear, high dimensional data, i.e., 10-bit parity (1024 pattern samples), 20-bit parity (more than one million pattern sample). These data are to be classified according to the proposed classification problems stated in this paper, i.e., HNP, ASP, HSNP, DHSNP, and/or PNP. Then, divide the bulk of data into three parts; training, cross validation, and testing. In doing this it is possible to measure the strategic methods used to initialize and train neural networks.

As for the classification of prime numbers, it is a very hard problem to learn with neural networks, finding better ways to teach neural networks may serve other fields that require the generation of large prime numbers efficiently. Neural networks should learn the classification task by looking into the bit patterns rather than using some special purpose algorithm whether it is an efficient or exhaustive search for prime number method. On using decimals from 1 to 1000 (some are prime numbers others are not) 83% of accuracy have been obtained for testing decimals with the ranges 64000 to 65000, and 74% of accuracy have been obtained for testing decimals

with the ranges 3000 to 4000. It is obvious that all training and testing sets used are disjoint. It is amazing how neural networks can learn to classify prime numbers from their binary representation, since in well structured algorithms the prime number should be non divisible by all decimals but itself and one. Reaching a classification rate of 74% or 83% is very promising, yet needs more investigation. It is worth mentioning that neural networks could not learn the prime classification problem using the direct decimal representations (as an Arabic numeral), therefore, it is our hope as we showed in the experimental results that the data is complex nonlinearity but solvable with neural networks. Training and testing of all the proposed data sets in this paper is left as a future work.

References:

[1] D.E. Rumelhart, and J.L.McClelland, *Parallel Distributed Processing*, Vol. 1, Cambridge, MA: MIT press, 1986.

[2] R.Setiono, On The Solution Of The Parity Problem By A Single Hidden Layer Feedforward Neural Network, *Neurocomputing* Vol.16, No.3, 1997, pp. 225-235.

[3] M. E. Hohil, D. Liu and S. H. Smith, Solving the *N*-bit parity problem using neural networks, *Neural Networks*, Vol.12, No.9, 1999, pp. 1321-1323.

[4] C.L.Giles, and C.W.Omlin, Pruning Recurrent Neural Networks for Improved Generalization Performance, *IEEE transactions on neural networks*, Vol.5, No.5, 1994, pp. 848-855.

[5] S. Hochreiter, and J. Schmidhuber, LSTN Can Solve Hard Long Time Lag Problems, In Mozer, M.C.,. Jordan, M.I, Petsche, T. eds., *Advances in Neural Information Processing Systems 9*, NIPS'9, Cambridge MA: MIT Press, 1997, pp. 473-479.

[6] D.Liu, M.E.Hohil, and S.H.Smith, N-bit parity neural networks:new solutions based on linear programming, *Neurocomputing*, Vol.48, 2002, pp. 477–488.

[7] E. Lavretsky, On The Exact Solution Of The Parity-*N* Problem Using Ordered Neural Networks, *Neural Networks*, Vol. 13, 2000, pp. 643–649.

[8] M.Z.Arslanov, D.U.Ashigaliev, and E.E.Ismail, N-bit parity ordered neural networks, *Neurocomputing*, Vol.48, 2002, pp. 1053 –1056.

[9] L. Franco, and S. A. Cannas, Generalization Properties of Modular Networks: Implementing the Parity Function, *IEEE transactions on neural networks*, Vol.12, No.6, 2001, pp.1306-1313.

[10] T. Nitta, Solving The XOR Problem And The Detection Of Symmetry Using A Single Complex-Valued Neuron, *Neural Networks*, Vol.16, No,8, 2003, pp. 1101-1105 .

[11] J. Elman, Finding Structure in Time, *Cognitive Science*, Vol.14, 1990, pp. 179-211.

[12] M.F. Mller, A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning, *Neural Networks*, Vol.6, No.4, 1993, pp.525-533.

[13] M.T. Hagan, H.B. Demuth, and M.H. Beale, *Neural Network Design*, Boston, MA: PWS Publishing, 1996.

[14] M. Al-Rawi, A Neural Network To Solve The Hybrid N-Parity: Learning With Generalization Issues, *Neurocomputing*, Vol.68, 2005, pp. 273-280.