

Constructive Proof of Efficient Pattern Storage in the Multi-layer Perceptron

Arunachalam Gopalakrishnan¹, Xiangping Jiang¹, Mu-Song Chen², and Michael T. Manry¹

¹Department of Electrical Engineering, University of Texas at Arlington, Arlington, Texas 76019

²Dept. of Electrical Engineering, Dai-Yeh Institute of Technology, Taipei, Taiwan

Abstract

In this paper, we show that the pattern storage capability of the Gabor polynomial is much higher than the commonly used lower bound on multi-layer perceptron(MLP) pattern storage. We also show that multi-layer perceptron networks having second and third degree polynomial activations can be constructed which efficiently implement Gabor polynomials and therefore have the same high pattern storage capability. The polynomial networks can be mapped to conventional sigmoidal MLPs having the same efficiency. It is shown that training techniques like output weight optimization and conjugate gradient attain only the lower bound of pattern storage. Certainly they are not the final solutions to the MLP training problem.

I. Introduction

Here we discuss the approximation capabilities and the efficiency of a three layer mapping neural network with polynomial or sigmoidal activation, in terms of the number of weights and the pattern storage. Some researchers have looked at the lower bound of the pattern storage[1,2]. Mu-Song Chen and Michael T. Manry[3] have modeled MLP using polynomial basis functions. In this paper, we use polynomial basis functions to decide on the bounds on the pattern storage capability of the MLP. Section II establishes the bounds on the pattern storage of the MLP and the equivalent Gabor polynomial. An efficient mapping technique to design a second and third degree network is given in sections III. Section IV shows a method to construct efficient multi-layer perceptron networks with sigmoidal activation. The words *weight* and *parameter* are used interchangeably in the discussions.

II. Bounds on Pattern Storage

An arbitrary N -ary Gabor polynomial can be written as

$$f(\mathbf{x}) = \mathbf{k}^T \cdot \mathbf{X} \quad (1)$$

where \mathbf{k} is the coefficient vector and \mathbf{X} is the column vector of the cardinal functions of the polynomial. Note that in neural network terminology, Gabor polynomials are known as functional link nets.[5] The width L of a Gabor polynomial is the number of terms or cardinal functions in the polynomial and is given by $L = (N+P)!/(N!P!)$ where P is the degree of the polynomial. Suppose there are N_v patterns \mathbf{x}_i ($1 \leq i \leq N_v$) with functional values of f_i ($1 \leq i \leq N_v$). Then the interpolating polynomial for the N_v patterns can be obtained by solving the system of equations,

$$\mathbf{f} = \underline{\mathbf{X}} \cdot \mathbf{k} \quad (2)$$

for \mathbf{k} , where the rows of the N_v by L matrix $\underline{\mathbf{X}}$ are comprised of vectors \mathbf{X}^T at different data points \mathbf{x}_i , and \mathbf{f} is the N_v by 1 vector of the function values.

Lemma 1. If the N_v patterns $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_v}$, contain a subset of L patterns which when expressed as cardinal functions of a Gabor polynomial are linearly independent, then $\underline{\mathbf{X}}^T \underline{\mathbf{X}}^{-1}$ exists ($\underline{\mathbf{X}}$ has full rank) and there exists a unique Gabor polynomial which can store L patterns.

A MLP with N inputs, 1 output, and N_h hidden units can be modeled using polynomial basis functions [6] as

$$f(\mathbf{x}) = \mathbf{w}_o^T \mathbf{C} \mathbf{X} \quad (3)$$

where \mathbf{w}_o is a 1 by N_h the output weight vector, \mathbf{C} is a $N+N_h+1$ by L matrix whose rows are the coefficients of the Gabor polynomial approximation of each input or hidden unit and $f(\mathbf{x})$ - function to be mapped. The number of patterns a MLP can store is the number of patterns it

can reproduce without error. The number of absolute

parameters in a network can be defined as the total number of weights or parameters in a network. The efficiency of the network can be defined as the ratio of number effective parameters to the width of the Gabor polynomial whose degree is equal to that of the network.

The bounds on pattern storage can be explained on the basis of this representation. The upper bound on the pattern storage depends on the number of effective parameters. Similarly the lower bound depends on the number of elements in the w_o vector.

Lemma 2. The multi-layer perceptron can store a minimum number of patterns, S_L equal to the number of output weights connecting to one output. i.e., the sum of number of hidden units, the number of inputs and the output threshold. This result is basically the same as that given in [1,2].

$$S_L = N + N_h + 1$$

Let N_w denote the number of weights and thresholds in an MLP which are on paths from any input to the first output.

Lemma 3. The upper bound S_u on the pattern storage capacity may be defined as the minimum of N_w and the width of the network's equivalent Gabor polynomial.

$$S_u = \min(L, N_w)$$

The actual pattern storage of an MLP is somewhere between that of its lower limit in *lemma 2* and its upper limit in *lemma 3*. If the lower bound in *lemma 2* turns out to be close to the correct value, then the MLP is likely to be superseded in the future by Volterra filters, functional link nets, and other networks which more directly implement Gabor polynomials. Our goal in this paper is to show that MLPs with sigmoidal activation can efficiently implement the Gabor polynomial, and therefore have high pattern storage capability.

We propose a special kind of three layer network with sparse connectivity for generating a given second degree or third degree polynomial. Regarding the quadratic and cubic mapping we state the following theorem.

Theorem 1. A second or third degree polynomial f can be realized with a three layer network efficiently in terms of the number of weights used, to an arbitrary accuracy.

In our discussions a constant weight of unity is not counted as a parameter for obvious reasons.

III. Mapping of Arbitrary Polynomials

A. 3-rd Degree Polynomial Mapping

An arbitrary N -ary 3-rd degree polynomial function can be efficiently represented by a

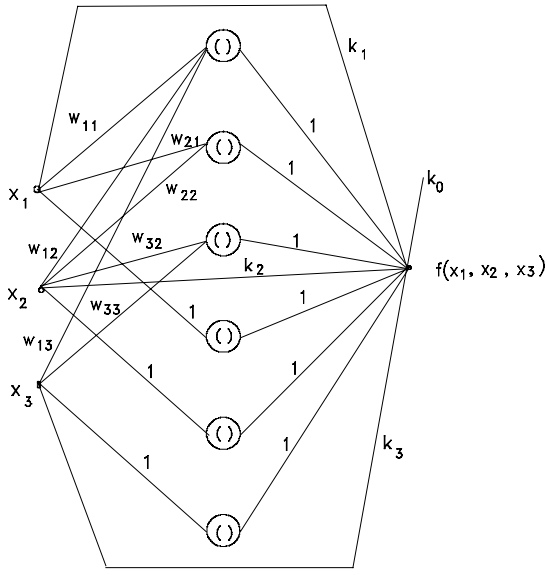
sparsely connected neural network (fig.1) with a single hidden layer with $N(N+1)/2$ units with a polynomial activation function of the form $bx^3 + ax^2$ where N units have one input, $N-1$ units have 2 inputs and so on, with all the inputs tied to the output and with a output threshold. The net function, the activation function and the output are respectively

$$\begin{aligned} net_k &= \sum_{i=1}^N w_{ki} x_i \\ f(net_k) &= a_k \cdot net^2 + b_k \cdot net^3 \quad (4) \\ f(\mathbf{x}) &= \sum_{i=1}^{N_h} f(net_i) \end{aligned}$$

Here, w_{ij} denotes the weight connecting the i -th hidden unit to the j -th input. The values of b_k coefficients are unity for $N_h - N$ units which are connected to more than one input. Thus N units have b_k coefficients, counted as free parameters. All the output weights are unity. The number of units N_h used for an N -ary 3-rd degree polynomial is $N_b = N(N+1)/2$. Let us take the example of mapping an arbitrary ternary 3-rd degree Gabor polynomial.

$$\begin{aligned} f(x_1, x_2, x_3) &= k_{111} x_1^3 + k_{222} x_2^3 + k_{333} x_3^3 + k_{112} x_1^2 x_2 \\ &+ k_{122} x_1 x_2^2 + k_{113} x_1^2 x_3 + k_{133} x_1 x_3^2 + k_{223} x_2^2 x_3 \\ &+ k_{233} x_2 x_3^2 + k_{123} x_1 x_2 x_3 + k_{11} x_1^2 + k_{22} x_2^2 \\ &+ k_{33} x_3^2 + k_{12} x_1 x_2 + k_{13} x_1 x_3 + k_{23} x_2 x_3 \\ &+ k_1 x_1 + k_2 x_2 + k_3 x_3 + k_0 \end{aligned}$$

This Gabor polynomial when mapped to the proposed structure results in the following set of equations which when solved in the given order, yields the values of weights.



$$() - ax^2 + bx^3$$

Figure 1. 3-rd degree function mapping

$$k_{113} = 3w_{11}^2w_{13}$$

$$k_{133} = 3w_{11}w_{13}^2$$

$$k_{123} = 6w_{11}w_{12}w_{13}$$

$$k_{112} = 3(w_{11}^2w_{12} + w_{21}^2w_{22})$$

$$k_{122} = 3(w_{11}w_{12}^2 + w_{21}w_{22}^2)$$

$$k_{223} = 3(w_{12}^2w_{13} + w_{32}^2w_{33})$$

$$k_{233} = 3(w_{13}^2w_{12} + w_{32}w_{33}^2)$$

$$k_{111} = b_4 + w_{21}^3 + w_{11}^3$$

$$k_{222} = b_5 + w_{32}^3 + w_{22}^3 + w_{12}^3$$

$$k_{333} = b_6 + w_{33}^3 + w_{13}^3$$

$$k_{13} = 2w_{11}w_{13}a_1$$

$$k_{12} = 2(w_{11}w_{12}a_1 + w_{21}w_{22}a_2)$$

$$k_{23} = 2(w_{12}w_{13}a_1 + w_{32}w_{33}a_3)$$

$$k_{11} = w_{11}^2a_1 + w_{21}^2a_2 + a_4$$

$$k_{22} = w_{11}^2a_1 + w_{22}^2a_2 + w_{32}^2a_3 + a_5$$

$$k_{33} = w_{13}^2a_1 + w_{33}^2a_3 + a_6$$

The linear terms and the constant term can be easily added to the equation by connecting the inputs directly to the output with appropriate gains and by introducing a threshold at the output respectively. Some results of the pattern storage capabilities of a third degree Gabor polynomial and that achieved with the proposed neural network structure are given in the table. Here, N_p denotes the number of patterns stored by the proposed network.

The mapping structure shown for third degree polynomial is efficient in terms of number of parameters or weights associated with them. In the network it can be seen that the number of weights is equal to the number of constants in the equivalent Gabor polynomial and they are capable of storing equal number of patterns. A similar network can be constructed for the second degree case also. These are not restricted to any definite number of inputs. Also, the result can be extended to multiple outputs where each output is defined by a different function.

| N | N_h | N_p | S_u | S_L |
|---|-------|-------|-------|-------|
| 2 | 3 | 10 | 10 | 5 |
| 3 | 6 | 20 | 20 | 10 |
| 4 | 10 | 35 | 35 | 15 |
| 5 | 15 | 56 | 56 | 21 |
| 6 | 21 | 84 | 84 | 28 |
| 7 | 28 | 120 | 120 | 36 |
| 8 | 36 | 165 | 165 | 45 |

Table. Pattern storage of MLP and a 3-rd degree polynomial

B. Higher Degree Polynomial Mapping

As the degree of the function is increased one should expect a similar kind of efficient mapping to exist. Surprisingly enough it does not. As the degree increases beyond three an efficient mapping of an arbitrary polynomial by a neural network structure fails and leads to

the following result.

Theorem 2: An efficient realization of a quartic polynomial using a single layer structure with monomial activation functions is impossible in the sense of number of free parameters associated with it.

This result is an extension of some of the facts established in geometry and linear algebra. Clebsch[7] proved geometrically that a general ternary quartic cannot be written as a sum of five fourth powers. In our terms, fifteen parameters do not suffice to represent any ternary quartic as a sum of powers because the fifteen polynomials in the fifteen variables are not algebraically independent. Sylvester[9] reformulated Clebsch's result. In effect he showed that if f is a sum of five fourth powers, then the 6 by 6 matrix associated to H_p , the psd quadratic form has less than full rank and so its determinant, the catalecticant, vanishes.

IV. Mapping Using Sigmoidal Units

In this section we show that the polynomial hidden unit of equation (4) can be implemented as a single sigmoidal unit. Every sigmoidal hidden unit realizes a square and cube term so that it can directly replace the units with polynomial activation.

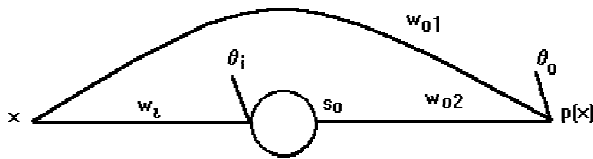


Figure 2. Sigmoidal approximation of degree-3 unit

Consider the sigmoidal network in fig.2. The hidden unit activation may be written as

$$f(net) = \sum_{m=0}^3 \frac{f^{(m)}(\theta_i)}{m!} (net - \theta_i)^m + R_3(net - \theta_i) \quad (5)$$

where $net = w_t x + \theta_i$, θ_i is the hidden unit threshold and $R_3(x)$ is the remainder. Equating $p(x)$ in fig.2 to $bx^3 + ax^2$, we see that a solution exists as long as the ratio $f^{(2)}(\theta)/f^{(3)}(\theta)$ varies from $-\infty$ to ∞ .

From fig. 3, we see that this condition is satisfied. The 3-degree network in fig.3 can therefore be realized with units having sigmoidal activations.

V. Examples

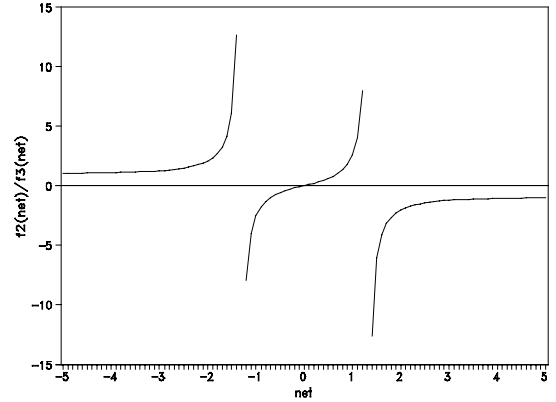


Figure 3. $f^2(net)/f^3(net)$ versus net for sigmoid

The mapping of an arbitrary third degree polynomial to an MLP is demonstrated in fig. 4. The MLP has one input, one hidden unit, and one output. The mapped network output is indistinguishable from the desired polynomial. The output of a network with same structure trained for 25000 iterations using back propagation is also given for comparison purposes. This demonstrates that a single sigmoid can replace the activation of (4).

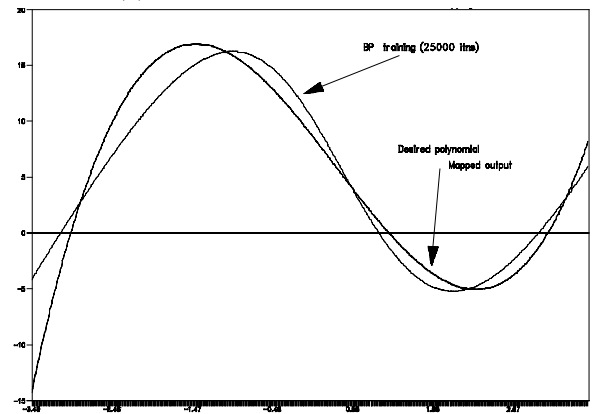


Figure 4. Sigmoidal modeling degree-3 polynomial

As a second example, we trained networks having topology 8-36-1, using varying numbers of random patterns. The mean square error of a network increases with the number of training patterns used. Fig.5 gives a comparison of the increase in error with the number of training patterns for different training algorithms, viz., back propagation(BP), output weight optimization(OWO) [10], conjugate gradient(CG)[11] and the proposed mapping method where an 8-variable 3-rd degree equivalent polynomial is used. Refer to the table for the bounds for

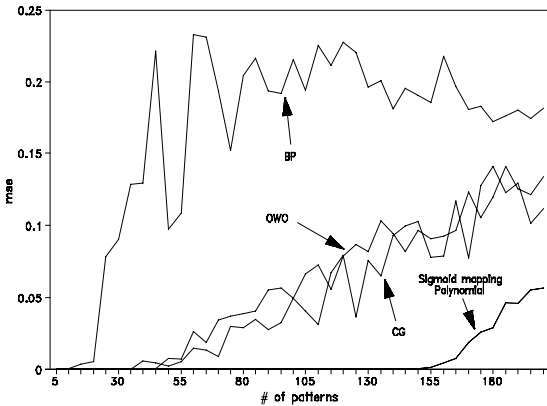


Figure 5. Training error versus number of patterns for four training algorithms

the 8-input case. The network was trained for 165 iterations using the three training algorithms to be consistent with the number of iterations used in the polynomial case which uses conjugate gradient algorithm to solve for the 165 parameters or weights. The plot of *mse* of the proposed direct mapping method remains arbitrarily close to zero up to 160 patterns reaching the upper bound of pattern storage (2). With back propagation the error curve breaks off at about 20 patterns and it may be improved to attain the lower bound(lemma 2) with larger number of iterations. The output weight optimization and conjugate gradient training algorithms are better than back propagation, as they have attained the lower bound of pattern storage. However even these algorithms lag far behind the Gabor polynomial and the proposed method. In fact, these fully connected networks have a much larger number of weights and thresholds, 359 in this case, than the 165 weights and thresholds used in the sparsely connected mapped network. These results indicate that the existing gradient training algorithms are not the final answer and certainly, better training algorithms could result in superior performance of the same network.

VI. Conclusions

It is clear that the proposed sparse network is more efficient than the normal MLP trained using back-propagation. Bounds on the pattern storage capabilities have been reviewed and the proposed networks are shown to be efficient in terms of pattern storage, and the number of parameters associated with them. It has been shown that an arbitrary second or third degree polynomial function can be directly mapped onto a MLP structure efficiently so as

to maximize the pattern storage capability. It has been shown that as the degree of mapping increases the MLP becomes less efficient than an equivalent Gabor polynomial in terms of pattern storage. The pattern storage results also indicate that back propagation results only in the lower bound of pattern storage. Though the output weight optimization and conjugate gradient techniques perform significantly better than BP, they also fall short of the upper bound. The mapping method shows a possibility and proves the existence of other training algorithms which would make the networks more efficient. The analysis can be extended to establish the efficiencies and pattern storage capabilities of higher degree networks.

Acknowledgement

This work was supported by NASA under Grant NAGW-3091 and by the NSF under grant IRI-9216545.

References

- [1] S-C Huang and Y-F Huang, " Bounds on the number of hidden neurons in multilayer perceptrons," *IEEE Transaction on Neural Networks*. Vol.2. No.1. pp. 47-55. Jan. 1991.
- [2] M.A. Sartori and P.J. Antsaklis, "A simple method to derive bounds on the size and to train multilayer neural networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 4, July 1991, pp. 467-471.
- [3] M.S. Chen and M.T. Manry, "Power series analysis of backpropagation neural networks", *Proc. IJCNN*, vol.I, 1991, pp 295-300.
- [4] M.S. Chen and M.T. Manry, "Backpropagation representation theorem using power series", *Proc. IJCNN*, vol.I, 1990, pp 643-648.
- [5] Yoh-Han Pao, *Adaptive pattern recognition and neural networks*, Addison-Wesley Publishing Company Inc., New York, 1989.
- [6] M.S. Chen and M.T. Manry, "Conventional modeling of the multilayer perceptron using polynomial basis functions", *IEEE Trans. on Neural Networks*, vol.4, No.1, Jan 1993, pp 164-166.
- [7] A. Clebsch, "Über Curven vierter Ordnung", *J. Reine Angew Math.* 59, 1861 pp 125-145.
- [8] E.W. Cheney, *Introduction to Approximation Theory*, Chelsea publications, 1982, New York.
- [9] J.J. Sylvester, "On a remarkable discovery in the theory of canonical forms and of hyperdeterminants", *Phil. Mag.* 2, 1851, pp 125-145. (Reprinted in "*Collected papers*", Cambridge University press, vol.1, 41, 1904, pp 265-283.)
- [10] M.T. Manry, X. Guan, S.J. Apollo, L.S. Allen, W.D. Lyle, W. Gong, "Output weight optimization for the multi-layer perceptron", *Proc. of 26th Asilomar conference on Signals, Systems and Computers*, 1992.
- [11] E. Barnard, "Optimization for training neural nets", *IEEE Transactions on Neural Networks*, Vol. 3, No.2, Mar 1992, pp 232-240.

