Turkish Journal of Physics

Volume 23 | Number 1 Article 5

1-1-1999

Neural Classifiers for Learning Higher-Order Correlations

Marifi GÜLER

Tr. J. of Physics
23 (1999) , 39 – 46.

Neural Classifiers for Learning Higher-Order Correlations

Marifi GÜLER
Department of Computer Engineering
Middle East Technical University
06531 Ankara - TURKEY

Received ...

Abstract

Studies by various authors suggest that higher-order networks can be more powerful and are biologically more plausible with respect to the more traditional multilayer networks. These architectures make explicit use of nonlinear interactions between input variables in the form of higher-order units or product units. If it is known a priori that the problem to be implemented possesses a given set of invariances like in the translation, rotation, and scale invariant pattern recognition problems, those invariances can be encoded, thus eliminating all higher-order terms which are incompatible with the invariances. In general, however, it is a serious setback that the complexity of learning increases exponentially with the size of inputs. This paper reviews higher-order networks and introduces an implicit representation in which learning complexity is mainly decided by the number of higher-order terms to be learned and increases only linearly with the input size.

1. Introduction

Biological evidence accumulated so far supports the presence of multiplicative-like operations in the brain which can be considered as the simplest most fundamental nonlinear operation [1]. It has been argued that such neural units represent an interesting and powerful set of computational machines [2-4].

The artificial neural units which have a linear combination of products of input components as input are called higher-order neurons or higher-order units. The input vectors are restricted to bipolar (i.e. $\{-1,+1\}$) components. Consider a network of M higher-order units with an N dimensional receptive field common to all units. The postsynaptic polarization potential for unit i ($i = 1,\ldots,M$) is given by

$$\phi_i(\mathbf{x}) = \omega_i^0 + \sum_j \omega_i^1(j)x_j + \sum_j \sum_k \omega_i^2(j,k)x_j x_k + \cdots + \sum_{j_1} \cdots \sum_{j_N} \omega_i^N(j_1,\ldots,j_N)x_{j_1}\ldots x_{j_N} \tag{1}$$

where $\mathbf{x}$ is the input pattern applied at the receptive field and $\omega_i^m(j_1, \ldots, j_m)$ is the weight connecting the product $x_{j_1} \ldots x_{j_m}$ to the i^{th} unit. The potential is passed through a non-linearity in order to compute the output of unit i as follows:

$$y_i(\mathbf{x}) = f(\phi_i(\mathbf{x}) \tag{2}$$

where f is usually taken as a sigmoidal function or the signum function. Each unit in this network can implement any bipolar function defined over the receptive field, provided that the weight values are set to appropriate values.

The implementational power of higher-order neurons is, however, overshadowed by the combinatorial explosion in the number of weights with the input size N. Therefore, feedforward networks and recurrent networks of higher-order neurons with a limited order (often second order) are studied [5-8]. On the other hand, if it is known a priori that the problem to be implemented possesses a given set of invariances like in the translation, rotation, and scale invariant pattern recognition problems, those invariances can be encoded, thus eliminating all higher-order terms which are incompatible with the invariances [2,9,10]. How to encode those invariances into the higher-order weights is the subject of Section II.

A major attraction of (artificial) neural networks is that for many problems they can be trained to perform as required on a variety of input data and, following training, can be expected to perform acceptably on data to which they have not been previously exposed. When a neural network performs acceptably on previously unseen input data, it is exhibiting a generalization capability. In many cases, the generalization error can be bounded (a worst-case analysis), and this bound can be made arbitrarily small by increasing the number of training instances. A useful bound can be established when the number of training instances exceeds the Vapnik-Chervonenkis (VC) dimension [11-13]. The VC dimension is a measure of the capacity of the family of classification functions realized by the learning machine.

The VC dimension of higher-order networks has been computed as the number of higher-order terms that the network employs [14]. Thus, for better generalization, higher-order networks should implement the training data using minimum number of higher-order terms possible. It is also a fact that in order to implement a certain function, in most cases only a few higher-order terms are required. Then, the storage of weights does not suffer from the combinatorial explosion problem; but this is not the case for the learning complexity, unless it is known a priori that the problem possesses some invariances, since it must be decided by the learning algorithm that whether a higher-order term is to be taken into consideration or not. Some approaches, in which learning complexity is decided by the number of higher-order terms in the minimal set, rather than the input size, have been proposed [3,15,16]. However, those approaches have their own problems such as they are very prone to the local minima problem. In Section III, we introduce a new model where the local minima problem is not that severe.

2. Invariant Pattern Recognition

Automatic pattern recognition is indispensable to modern systems dealing with advanced information processing. It is desirable that such systems had the additional property of recognizing patterns, without being influenced by distortions or transformations. For example, in the recognition of handwritten letters, our interest is the structure within the letter rather than the size or location or orientation of that letter. Invariant pattern recognition is one of the hardest families of problems in the theory of perception and in computer vision. The problem of invariant recognition is to understand how our perception of an object remains unaffected in spite of the considerable changes that the retinal image of the object undergo [17].

For the practical application of neural networks, there is a need for models that exhibit invariance to certain transformations in object recognition. Among the most important of these transformations are translation, scale, and rotation. We are going to discuss, here, the theoretical justification of encoding those invariances into the higher-order networks. The network model used is the model described by Eqn. (1), and the receptive field is the input pattern, say picture of a car, on a two-dimensional grid of black and white pixels. Further details can be found in [2,9,10,18].

0.1. Translation Invariance

Consider a translation of the input pattern x_j by n positions so that

$$y_i(x_{j+n}) = y_i(x_j) \tag{3}$$

int terms of the notation of Eqn. (2). Since $\mathbf{x}$ can be any pattern, this imposes term by term equality in the argument of the function f. Hence, we have from Eqn. (1) for the first and second order terms

$$\sum \omega_i^1(j)x_j = \sum \omega_i^1(j)x_{j+n} \tag{4}$$

$$\sum\sum \omega_i^2(j,k)x_j x_k = \sum\sum \omega_i^2(j,k)x_{j+n}x_{k+n} \tag{5}$$

Making the substitutions $j \to j-n$, $k \to k-n$ yields

$$\sum \omega_i^1(j)x_j = \sum \omega_i^1(j-n)x_j \tag{6}$$

$$\sum\sum \omega_i^2(j,k)x_j x_k = \sum\sum \omega_i^2(j-n,k-n)x_j x_k \tag{7}$$

It can be verified that, if the limits of the summations on the right-hand side are infinite or if a grid with periodic boundary conditions is considered, the limits can be set equal on both sides. Then, it follows that

$$\omega_i^1(j) = \omega_i^1(j-n) \tag{8}$$

$$\omega_i^2(j,k) = \omega_i^2(j-n,k-n) \tag{9}$$

These equations imply that the first order weights are independent of the input position, whereas the second order weights are functions of vector differences only:

$$\omega_i^1(j) = \omega_i^1 \tag{10}$$

$$\omega_i^2(j,k) = \omega_i^2(j-k) \tag{11}$$

These requirements reduce the number of second order weights from N^2 to order N. In most cases, the second order terms are sufficient to implement translational invariance.

0.2. Scale Invariance

The condition to be satisfied in case of scale invariance is the following:

$$y_i(x_{aj}) = y_i(x_j) \tag{12}$$

where a is a scale factor. Applying the same procedure as above leads to the following constraints on the weights:

$$\omega_i^1(j) = \omega_i^1(j/a) \tag{13}$$

$$\omega_i^2(j,k) = \omega_i^2(j/a, k/a) \tag{14}$$

It is appropriate to consider a scale transformation in rectangular coordinates (u, v), so that position j is expressed as the vector $j = (u_j, v_j)$. A set of solutions to the second order weight constraint is

$$\omega_i^2(u_j, v_j, u_k, v_k) = \omega_i^2(u_j/v_j, u_k/v_k) \tag{15}$$

$$\omega_i^2(u_j, v_j, u_k, v_k) = \omega_i^2(u_j/u_k, v_j/v_k) \tag{16}$$

$$\omega_i^2(u_j, v_j, u_k, v_k) = \omega_i^2((u_j - u_k)/(v_j - v_k)) \tag{17}$$

Any other solution that satisfies Eqn. (14) could also be used.

0.3. Rotation Invariance

In case of rotation about the origin, we consider a two-dimensional space in polar coordinates (r, θ). It can be shown that the following weight constraints fulfill the requirement for rotational invariance:

$$\omega_i^1(r_j, \theta_j) = \omega_i^1(r_j) \tag{18}$$

$$\omega_i^2(r_j, r_k, \theta_j, \theta_k) = \omega_i^2(r_j, r_k, \theta_j - \theta_k) \tag{19}$$

0.4. Combined Transformations

Combination of transformations, e.g. scale and translation invariance, is interesting from applications point of view. Consider first the case of translation by n positions, followed by a change of scale by factor a. The constraints on the first and second order weights are

$$\omega_i^1(j) = \omega_i^1((j-n)/a) \tag{20}$$

$$\omega_i^2(j,k) = \omega_i^2((j-n)/a, (k-n)/a) \tag{21}$$

while for scale followed by translation the constraints are

$$\omega_i^1(j) = \omega_i^1(j/a - n) \tag{22}$$

$$\omega_i^2(j,k) = \omega_i^2(j/a - n, k/a - n) \tag{23}$$

If we consider the case in rectangular coordinates (u, v), the solution

$$\omega_i^2(u_j, v_j, u_k, v_k) = \omega_i^2((u_j - u_k)/(v_j - v_k)) \tag{24}$$

satisfies both scale and translation invariance independent of the order.

In case of a change of scale by a factor a and rotation about the origin by an amount α for a system in polar coordinates (r, θ), the order of transformation makes no difference. The weight constraints for the first and second order weights are

$$\omega_i^1(r_j, \theta_j) = \omega_i^1(r_j/a, \theta_j - \alpha) \tag{25}$$

$$\omega_i^2(r_j, r_k, \theta_j, \theta_k) = \omega_i^2(r_j/a, r_k/a, \theta_j - \alpha, \theta_k - \alpha) \tag{26}$$

The first order weight is independent of the input, whereas a convenient solution to the second order constraint is

$$\omega_i^2(r_j, r_k, \theta_j, \theta_k) = \omega_i^2(r_j/r_k, \theta_j - \theta_k) \tag{27}$$

Simultaneous invariance with respect to translation, scale, and rotation can be achieved using third order weights. The weights $\omega_i^3(j,k,l)$ and $\omega_i^3(j^{'}, k^{'}, l^{'})$ are set equal if the corresponding triangles (j,k,l) and $(j^{'}, k^{'}, l^{'})$ have the same angles.

0.5. Learning algorithm

After the imposition of the transformational invariances, the values of the independent weights are to be decided through a learning algorithm. The simple perceptron learning rule can be used, where each weight is updated as

$$\Delta\omega_i^m(j_1, \ldots, j_m) = \eta(d_i(\mathbf{x}) - y_i(\mathbf{x}))x_{j_1} \ldots x_{j_m} \tag{28}$$

where $d_i(\mathbf{x})$ is the desired output for the input image $\mathbf{x}$ and η is the step size. Considering the bipolar nature of the inputs, the above perceptron rule is efficient and can lead to

rapid convergence. Other learning algorithms based on the minimization of the cost function

$$E = \frac{1}{2} \sum_i \sum_{\mathbf{x}} (d_i(\mathbf{x}) - y_i(\mathbf{x}))^2 \tag{29}$$

through the gradient-descent method or some other methods, can be used.

3. Learning Higher-Order Terms in an Implicit Represantation

In the general case, i.e. when there exists no a priori known invariances in the problem, the main objective is how to learn those relevant higher-order terms in an algorithm with polynomial time increase with the input size. A well known example that satisfies this objective is the network of product units [3]. However, there is a serious set back. The update equations in the learning algorithm contain periodic functions, which results in a severe suffering from the local minima problem.

We introduce, here, a network model where learning complexity increases linearly with the input size and no periodic functions are present. Only the main points of the model are stated here. Further details and simulation results are to be discussed somewhere else [19]. The network implements type of functions $F : \{-1, 1\}^N \rightarrow \{-1, 1\}^M$, where $M = 1$ is taken for simplicity. It is, however, straightforward to extend it to the case where M is any desired positive integer.

The network output for the input pattern $\mathbf{x}$, denoted by $O(\mathbf{x})$, is given as

$$O(\mathbf{x}) = tanh(\beta \sum_{m=1}^{K} g_m A^m(\mathbf{x})) \tag{30}$$

where β is a constant, K is a measure of the network's topological complexity analogous to the number of hidden units in a feedforward multilayer network, and g_m are adjustable weights to be determined by the learning algorithm. A^m are sums of some Gaussians:

$$A^m(\mathbf{x}) = \sum_{l=q(\mathbf{x})}^{z(\mathbf{x})} (-1)^{l-l_i} exp(-(\alpha^{lm} B^{lm}(\mathbf{x}))^2) \tag{31}$$

where α^{lm} are adjustable weights to be determined by the learning algorithm, l_i is the smallest integer greater than $-(N + 5)/2$, $q(\mathbf{x})$ is the smallest integer greater than $-(n^-(\mathbf{x}) + 5)/2$ and $z(\mathbf{x})$ is the largest integer less than $(n^+(\mathbf{x}) + 4)/2$; where $n^-(\mathbf{x})$ and $n^+(\mathbf{x})$ are the number of -1's and the number of $+1$'s in the input pattern $\mathbf{x}$, respectively. B^{lm} are defined as

$$B^{lm}(\mathbf{x}) = \sum_{i=1}^{N} L(\gamma \omega_i^m) x_i - L(\gamma v^m) - 2l \tag{32}$$

where γ is a constant, ω_i^m and v^m are adjustable weights to be determined by the learning algorithm and L denotes the logistic function, defined by

$$L(x) = \frac{1}{1 + exp(-x)} \tag{33}$$

A^m can implement any higher-order term in Eqn. (1). Consider the higher-order term $x_{k_1}x_{k_2}\ldots x_{k_r}$, where k_r is even. Let

$$w_i^m = \begin{cases} +\infty & \text{for} \quad i = k_1, k_2, \ldots, k_r \\ -\infty & \text{otherwise} \end{cases}$$

and $v^m = -\infty$. This implies that

$$B^{lm}(\mathbf{x}) = x_{k_1} + x_{k_2} + \cdots + x_{k_r} - 2l \tag{34}$$

There exists l^* so that $B^{lm}(\mathbf{x}) = 0$ at $l = l^*$, hence all contributions in the sum in $A^m(\mathbf{x})$ are negligible except the one at $l = l^*$. Therefore, $|A^m(\mathbf{x})|$ is approximately 1 and the overall sign can be absorbed into g_m if required. The above higher-order term and $A^m(\mathbf{x})$ subject to $B^{lm}(\mathbf{x})$ in Eqn. (34), have the same following symmetries. The first symmetry is the invariance with respect to the interchange of the values of x_{k_i} and x_{k_j}, where $1 \leq i \leq r$ and $1 \leq j \leq r$. The second symmetry is that when the sign of x_{k_i}, where $1 \leq i \leq r$, is reversed then both the higher-order term and the $A^m(\mathbf{x})$ reverse their signs.

Implementation of the higher-order term $x_{k_1}x_{k_2}\ldots x_{k_r}$ in case of odd r can be seen by taking $v^m = +\infty$ and following the above steps.

Learning of the adjustable weights is through the minimization of the following cost function by means of the gradient-descent method:

$$E = E_1 + E_2 \tag{35}$$

where the first part is the usual quadratic cost

$$E = \frac{1}{2}\sum_{\mathbf{x}}(d(\mathbf{x}) - O(\mathbf{x}))^2 \tag{36}$$

and the second part performs weight elimination [20]

$$E = \frac{1}{2}\lambda \sum_{m=1}^{M} \frac{(g_m/g^*)^2}{1 + (g_m/g^*)^2} \tag{37}$$

where λ and g^* are some constants. The role played by the weight elimination is to make those unnecessary g weights vanish, so that implementation takes place using minimal number of higher-order terms.

4. Conclusions

The use of higher-order neurons has been examined in this paper. Invariant recognition of patterns and images using higher-order neural networks has been reviewed.

A model for learning higher-order correlations within a given data has been proposed. In the model, learning complexity is basically decided by the number of higher-order terms to be implemented and increases linearly with the input size. The model should benefit from a broad range of potential applications.

References

[1] C. Koch and T. Poggio, In *Single Neuron Computation*, Eds. T. McKenna, J. Davis and S.F. Zornetzers (Academic Press, 1992) 315.

[2] C.L. Giles and T. Maxwell, *Applied Optics* **26** (1987) 4972.

[3] R. Durbin and D.E. Rumelhart, *Neural Computation* **1** (1989) 133.

[4] N.J. Redding, A.. Kowalczyk and T. Downs, *Neural Networks* **6** (1993) 997.

[5] Y.C. Shin and R. Sridhar, *Proc. WCNN'93* **1** (1993) 585.

[6] P. Peretto and J.J. Niez, *Biol. Cybern.* **54** (1986) 53.

[7] G.A. Kohring, *J. Phys. France* **51** (1990) 145.

[8] E. Gardner, *J. Phys. A: Math. Gen.* **20** (1987) 3453.

[9] S. Perantonis and P. Lisboa, *IEEE Trans. Neural Networks* **3** (1992) 241.

[10] L. Spirkovska and M.B. Reid, *IEEE Trans. Neural Networks* **4** (1993) 276.

[11] V.N. Vapnik and A.Y. Chervonenkis, *Theory of Probability and Its Appl.* **16** (1971) 264.

[12] V.N. Vapnik, *Estimation of Dependences Based on Empirical Data* (Springer-Verlag, 1982).

[13] Y.S. Abu-Mostafa, *Neural Computation* **1** (1989) 312.

[14] S. Young and T. Downs, *Electronics Lett.* **29** (1993) 1491.

[15] M. Güler and E. Şahin, *Proc. WCNN'94* **3** (1994) 730.

[16] M. Güler and E. Şahin, *Proc. TAINN* (1996) 15.

[17] R. Duda and P. Hart, *Pattern Classification and Scene Analysis* (John Wiley, 1973).

[18] G.L. Giles, R.D. Griffin and T. Maxwell, *Neural Information Processing Systems*, American Inst. of Phys. Conf. Proc. (1988).

[19] M. Güler, in preparation.

[20] A.S. Weigend, D.E. Rumelhart and B.A. Huberman, *Proc. 1990 Connectionist Models Summer School* (1990) 65.