Indian Journal of Science and Technology, Vol 7(5), 622–627, May 2014
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Neural Classification of *h*- and *p*-version Elements

Z. Goudarzi[1] and A. Abedian[2*]

[1]Department of Computer Engineering, University of Isfahan, Isfahan, Iran.
[2]Department of Mechanical Engineering, Daneshpajoohan Higher Education Institute, Isfahan, Iran; abedian@me.iut.ac.ir

Abstract

This paper deals with a comparative performance of traditional elements and high order elements, making use of their formulation as vectors (or patterns) in a multi-dimensional space of proper attributes. The classification can be carried out with the help a self-organizing feature map of Kohonen with the patterns corresponding to the input space. The work makes use of the four attributes: its number of nodes, number of Lejendre terms, maximum degree of interpolation polynomials and number of degrees of freedom per node, though a more general characterization is also possible.

Keywords: Classification, Finite Elements, Kohonen's Network, Neural Networks

1. Introduction

The finite element method is usually applied in order to find approximations of complex boundary value problems arising in many engineering disciplines. These approximate solutions then serve as a basis for decisions which have to be made, for instance, during a design process in civil or mechanical engineering. Since the finite element method is numerical, it involves different sources of errors and a reliable decision can therefore only be made if the error is somehow controlled. For the elements which need to be refined, a decision has to be made whether the polynomial degree should be increased (*p*-version) or the element size should be reduced (*h*-version)[1,2]. High order elements are used in the *p*-version FEM and low order element are used in the *h*-version FEM, therefore it is important to know relationship between these elements. Also, still finite element performance lacks a sufficient quantitative characterization. That is why investigations into similarity among finite elements or their "closeness" and, subsequently, their classification may be useful. Due to the great variety of finite elements available[3,4], this seems particularly relevant. The work thus focuses on a comparative performance of finite elements, instead of its common evaluation in terms of approximation theories. A traditional classification of finite elements is based on a single attribute and it is not satisfactory. For example, by their dimensionality, the finite elements are divided into three trivial classes: linear, plane and 3Delements[5]. Beltzer[6] introduced a classification for traditional elements based on dimensionality, number of nodes, polynomial degree and number of degrees of freedom per nodes. But total classification of elements such as 1D, 2D and 3D may not be useful since it is impossible to use them instead of each others. Therefore, in the present study, the elements with the same dimensionality were classified.

The interest in such a classification is not of a theoretical nature only. Since the elements, which belong to the same class, should show a similar performance, the classification is of an obvious applied value too. The present work proposes a general approach to this problem, which is based on application of neural networks.

2. SOM Network for Finite Elements

Let Z be the number of parameters (attributes) for a set J of elements, which define the finite element, and state

*Author for correspondence

these as a vector X in a Z-dimensional space, $X = (X_1, X_2, ..., X_Z)$, $X \in J$.

Thought many parameters are possible to use for classification, but here just the most important of them were used. Under above, the basic attributes for spatial dimension are: its number of nodes N, the maximum degree of approximating polynomials M, number of Lejendre terms L, and the number of degrees of freedom per node F. Thus, for the present case $Z = 4$, the X-vector being specified as $X = (N, M, L, F)$.

Consider the above set J of k finite elements specified by their vectors, X_i, $i = 1, 2, ..., k$. As far as the problem and mesh-dependent effects are excluded from consideration, the vector X_i controls the performance of the ith element. Therefore, elements, whose vectors X_i are close to one another (belong to the same class or cluster), should show similar performance.

Since $J = \{X_i, i = 1, 2, ..., k\}$ may be considered as a set of patterns in a four-dimensional space, one may invoke the neural network paradigm for classification (clustering) purposes.

The reader is referred to Hagan et al.[7] and Haykin[8] for comprehensive expositions of neural networks. Among various neural networks, the so-called self organizing feature map (SOM) of Kohonen is of particular interest[9,10]. One of the most interesting aspects of SOM is that they learn to classify data without supervision.

In order to evaluate the activity bubbles of biological systems, without having to implement the nonlinear on-center/off-surround feedback connections, kohonen designed the following simplification. His Self-Organizing feature Map (SOM) network first determines the winning neuron i^* using the same procedure as the competitive layer. Next, the weight vectors for all neurons within a certain neighborhood of the winning neuron are updated using Kohonen rule,

$$_iW(q) = {}_iW(q-1) + \alpha(P(q) - {}_iW(q-1)),$$

where w is weight matrix, α is the learning rate, P is the input vector, $i \in N_{i^*}(d)$ and $N_{i^*}(d)$ contains the indices for all of the neurons that lie within a radius d of the winning neuron i^*:

$$N_i(d) = \{j, d_{ij} \le d\}$$

When a vector P is presented, the weights of the winning neuron and its neighbors will move toward P. The result is that, after many presentations, neighboring neurons will have learned vectors similar to each other.

A simple SOM is shown in Figure 1. This network may learn both the distribution and topology of a set of input patterns. At the end of the learning process by Kohonen's rule, the neurons become selectively tuned to classes of input patterns. This amounts to the formation of a map, which correlates spatial locations of the neurons in the output layer (lattice) with intrinsic features of the input patterns.

3. Classification of 1D Finite Elements

Assume that the set J consists of fourteen 1D-elements ($k = 14$) given in Table 1. Let us treat this problem with the help of a SOM. For all the elements of the set the dimensionality are equal one. The pattern vector,X_i, $i = 1, 2, ..., 14$, becomes four dimensional, $Z = 4$, implying $X_i = (N_i, M_i, L_i, F_i)$. The output layer may be arranged as a two-dimensional lattice and the number of output neurons, S, which depends on a desirable number of classes (or clusters), may be taken as $S = 2 \times 2$ or $S = 2 \times 3$. The learning process starts with random initialization of weights for each of the output neurons. Then, the algorithm picks in a random fashion an input vector $X_i = (N_i, M_i, L_i, F_i)$, $i = 1, 2, ..., 14$ from Table 1 and presents it to the network. Each time the vector is presented, a neuron wins a competition, as mentioned in the above. According to the learning rule by Kohonen, this neuron and its neighbors move their weight vectors even closer to the input vector. This process has two tendencies: first, the weight vectors spread out over the input space as the algorithm picks a new input pattern; second, the weights of adjacent neurons move in the same direction. The learning process, which is, in fact, specification of the weight matrix, W, usually goes on for 100 iterations or more, until the map reaches a steady state.

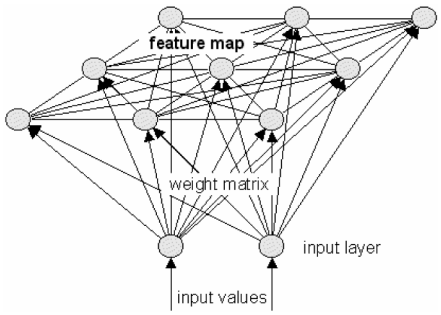

Figure 1. Principle of a SOM network.

Table 1. Set of 1D elements

Finite element	Identifying number,#	Number of nodes, N	Polynomial degree, M	Number of Legendre terms, L	Degrees of freedom per node, F
2-node linear	1	2	1	0	3
3-node parabolic	2	3	2	0	3
4-node cubic	3	4	3	0	3
5-node quadratic	4	5	4	0	3
2-node cubic–cubic arch	5	2	3	0	4
2-node Quantico–Quantico arch	6	2	5	0	6
2-node beam	7	2	3	0	2
2-node, L1	8	2	2	1	3
2-node, L2	9	2	3	2	3
2-node, L3	10	2	4	3	3
2-node, L4	11	2	5	4	3
2-node, L5	12	2	6	5	3
2-node, L6	13	2	7	6	3
2-node, L7	14	2	8	7	3

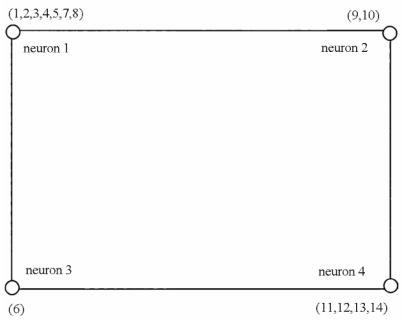

Figure 2. Classification by four neuron 1D elements.

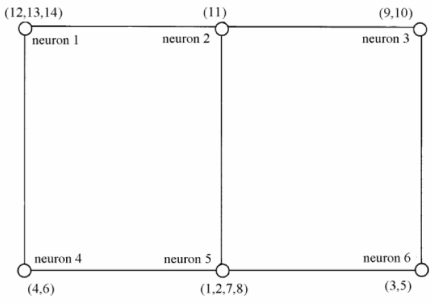

Figure 3. Classification by six neuron 1D elements.

Figures 2 and 3 show the results, computed using MATLAB software neural network toolbox, referring to a finite element by its identifying number (in brackets) as given in the second column of Table 1.

As shown in Figure 2, the first neuron selects the finite elements #1, #2, #3, #4, #5, #7 and #8, the second neuron selects the elements #9 and #10, the third neuron selects the elements #6 and the fourth neuron #11, #12, #13, #14. Therefore just element #8 of high order elements has classified in same class as low order elements. For seeing more precisely elements classified as Figure 3 that shows again just element #8 gathered by traditional elements. However, here the other lower order and high order elements have classified more precisely. Tables 3 and 4 present the weight matrices of the two networks, which are $(4 \cdot 4)$ and $(6 \cdot 4)$, respectively, arrived at by the Kohonen algorithm.

Table 2. Weight matrix of four–neuron classification network

2.6507	2.7547	0.48931	3.111
2.365	3.7761	1.9582	3.1422
2.399	3.8874	1.7609	3.3293
2	5.7756	4.4456	3.2475

Table 3. Weight matrix of six–neuron classification network

2	6.2146	5.2146	3
2.4043	5.3034	3.2195	3.5097
2.3962	4.0561	1.9439	3.302
2.9121	4.1861	0.78779	4.0673
2.3061	2.41	0.52872	2.8706
2.6348	3.1256	0.4266	3.4624

4. Classification of 2D Finite Elements

Table 2 shows a set of 2D traditional and high order elements. Since the input patterns are defined in 4D-space of the attributes, resort to a one-dimensional output lattice may cause information losses. So, the output layer is arranged as a two-dimensional lattice. With $k = 17$, a net should consist of several output neurons only. Below we first investigate a network with $S = 2 \times 2$ neuron

lattice and then a network with $S = 2 \times 3$ lattice, which may be referred to as the first- and second-order classification networks, respectively.

The learning process starts with random initialization of four weights for each of the output neurons and then evolves in a way described in the above. After 100 iterations, each of the finite elements given in Table 4 becomes associated with one of the neurons. Figures 4 and 5 graphically display the results for $S = 2 \times 2$ and $S = 2 \times 3$, respectively. The identifying numbers of the finite elements selected by a particular neuron are shown in brackets. As shown in Figure 4 just element #10 of high order elements has classified in same class with low order elements. For seeing more precisely elements classified as Figure 5 that shows again just element #10 gathered by traditional elements and increasing number of neurons did not help network to classify another high order element to contribute in low order class, therefore it just classifies other elements more precisely. Also there is another concept "migration" which can happen by rising number of output neurons that an element tends to another neuron and increase or decrease number of element f each neuron. This migration cause to elements classified more precisely.

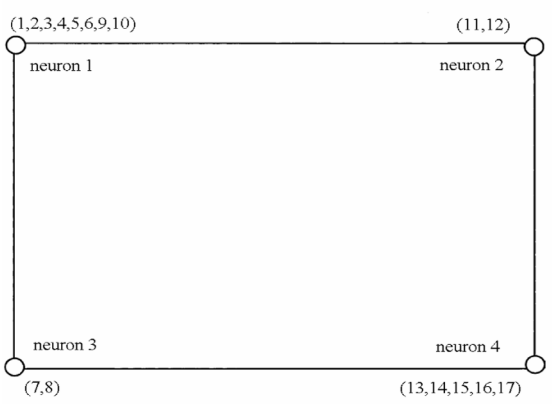

Figure 4. Classification by four neuron 2D elements.

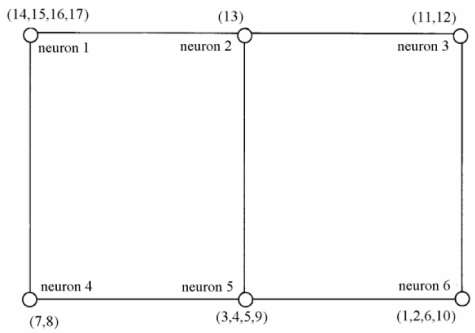

Figure 5. Classification by six neuron 2D elements.

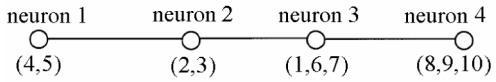

Figure 6. Classification by four neuron 3D elements.

5. Classification of 3D Finite Elements

As shown in Figure 6, the first neuron selects the finite elements #4 and #5, the second neuron selects the elements #2 and #3, the third neuron selects the elements #1, #6 and #7 and the fourth neuron #8, #9 and #10. Therefore in neuron 3 are contribution of low order and high order elements (Table 5).

Because of similarity of 3D elements in number of nodes this effect has ignored and classification applied with other three inputs.

The network has distributed the finite elements with respect to four classes or clusters, taking into account their "closeness". The network also preserves

Table 4. Set of 2D elements

Finite element	Identifying number,#	Number of nodes, N	Polynomial degree, M	Number of Legendre terms, L	Degrees of freedom per node, F
3-node plane triangle	1	3	1	0	2
4-node plane quadrilateral	2	4	2	0	2
6-node plane triangle	3	6	2	0	2
6-node plane quadrilateral	4	6	3	0	2
8-node plane quadrilateral	5	8	3	0	2
4-node rectangular plate	6	4	4	0	3
4-node rectangular shell	7	4	6	0	6
4-node cylindrical shell	8	4	6	0	12
10-node plane triangle	9	10	3	0	2
4-node quadrilateral, L1	10	4	3	1	2
4-node quadrilateral, L2	11	4	4	2	2
4-node quadrilateral, L3	12	4	5	3	2
4-node quadrilateral, L4	13	4	6	4	2
4-node quadrilateral, L5	14	4	7	5	2
4-node quadrilateral, L6	15	4	8	6	2
4-node quadrilateral, L7	16	4	9	7	2
4-node quadrilateral, L8	17	4	10	8	2

Table 5. Set of 3D elements

Finite element	Identifying number,#	Number of nodes, N	Polynomial degree, M	Number of Legendre terms, L	Degrees of freedom per node F
4-node tetrahedron	1	4	1	0	3
8-node brick	2	8	3	0	3
9-node wedge	3	9	3	0	3
10-node parabolic tetrahedron	4	10	2	0	3
15-node parabolic wedge	5	15	3	0	3
4-node tetrahedron, L1	6	4	2	1	3
4-node tetrahedron, L2	7	4	3	2	3
4-node tetrahedron, L3	8	4	4	3	3
4-node tetrahedron, L4	9	4	5	4	3
4-node tetrahedron, L5	10	4	6	5	3

the topology of the input set, for example indicating that the third class is closer to the second class and fourth class.

6. Conclusion

The work focuses on a comparative performance of high order and traditional finite elements, instead of its evaluation in terms of interpolation theories. The finite element is formulated as a vector or pattern in a multi-dimensional space of its attributes. Excluding from consideration the problem and mesh-dependent effects, this work makes use of a four-dimensional characterization: its number of nodes, maximum degree of interpolation polynomials, number of legendre terms and number of degrees of freedom per node.

The pattern classification may be carried out with the help a SOM of Kohonen. These networks learn both the distribution and topology of a set of input patterns.

The classification is of an obvious applied value, as the elements, which belong to the same class, should show a similar performance. Also suitability of neural network for evolution of high order and low order elements proved.

7. References

1. Szabo BA, Babuska I. Finite element analysis. John Wiley & Sons; 1991.
2. Babuska I, Szabo BA, Katz IN. The *p*-version of the finite element method. SIAM J Numer Anal. 1981; 18(3):515–45.
3. MacNeal RH. Finite elements: their design and performance. New York: Marcel Dekker; 1994.
4. Beltzer A. Engineering analysis with Maple/Mathematica. London: Academic Press; 1995.
5. Kardestuncer H, Norrie DH. Finite element handbook. New York: McGraw-Hill, Inc.; 1987.
6. Beltzer A, Sato T. Neural classification of finite elements. Comput Struct. 2003; 81:2331–35.
7. Hagan MT, Demuth HB, Beale M. Neural networks design. Boston: PWS Publ. Co; 1995.
8. Haykin S. Neural networks. Upper Saddle River, NJ: Prentice–Hall; 1999.
9. Kohonen T. Self-organized formation of topologically correct feature maps. Biol Cybern. 1982; 43(1):59–69.
10. Kohonen T. The self-organizing map. Proc of the IEEE. 1990; 78(9):1464–80.