A hierarchical Self-Organizing Map for classification problems

Bart De Ketelaere[(1)], Dimitrios Moshou, Peter Coucke, Josse De Baerdemaeker

Department of Agro-Engineering and Economics, Laboratory of Mechanical Engineering, K.U. Leuven, Kardinaal Mercierlaan 92, 3001 Heverlee, BELGIUM
[(1)]tel: +32-16-321478, fax: +32-16-321994, email: bart.deketelaere@agr.kuleuven.ac.be

Abstract

A hierarchical Self-Organizing Map has been developed for solving classification problems, where several measurements have been taken from one object. The algorithm will be explained with the use of the practical example where it is the aim to classify eggs according to their shell state. Broken eggs will be separated in this way from intact eggs. The classification architecture actually consists of two different SOMs. The first SOM clusters the data in an unsupervised way. Afterwards, the ordered activations of each object are collected and fed to the second SOM which actually clusters the ordered activations and assosiates them with a class, which is presented as a binary vector. This class-vector is assigned to every node in the second map as an output weight and it is learned with Kohonen's learning rule.

I. Introduction

The Self-Organizing Map (Kohonen, 1995) is a useful tool to represent high-order data in a two-dimensional discrete lattice of neurons. In this step, information is kept about the place of the input in the input-space, so that inputs who are near in the input-space are projected near each other in the map. However, the original SOM is an unsupervised clustering method. Some adjustments have to be made to make it supervised in a first step. Furthermore, a basic SOM can't combine several measurements from one object in the case where there is no way to order the observations so that a distance metric can be used reliably. This ordering is necessary in order to label the collected observations with only one class. How these two restrictions are avoided, will be explained in the following text. Previous attempts to use SOM as a supervised learning machine include (Ichiki et al., 1993) who use output weights to represent the classes in the same vector as the inputs and cluster with SOM these mixed vectors and (Ritter et al., 1992)

II. The hierarchical SOM

The Self-Organizing Map (Kohonen 1995) is a neural network that maps signals ($\mathbf{x}$) from a high-dimensional space to a one- or two-dimensional discrete lattice of neuron units ($\mathbf{s}$). Each neuron stores a weight ($\mathbf{w_s}$). The map preserves topological relationships between inputs in a way that neighboring inputs in the input space are mapped to neighboring neurons in the map space. The learning algorithm for the input and output weights is derived from the original Kohonen algorithm (Kohonen 1982) and has as:

$$\Delta \mathbf{w_s} = \varepsilon h(\mathbf{x}-\mathbf{w_s})$$

where ε and h are the learning rate and the neighborhood kernel respectively. The neighborhood kernel used had the form of a Gaussian like:

$$h = e^{-\frac{\|X - W_s\|^2}{2\sigma^2}}$$

where ≡.≡ denotes the Euclidean norm and σ denotes the variance of the Gaussian distribution. Starting from the basic SOM, the first thing to do is to make it supervised. In the practical case of the eggs, three classes are obtained: class one contains all the intact eggs, class two all the eggs that have a crack that is not opened (no eggwhite is leaking out, the membrane is intact) and class three contains the eggs with an open breakage. There is a need to separate class two from class three because the destiny of the eggs from both classes are different. Where the eggs of class two can be used for a pasteurization process, the eggs that belong to class three have no economic use and will be eliminated as soon as possible. This has the advantage that the sorting machine will not be spoiled with eggwhite. Instead of presenting the classes like they are (1, 2 and 3), they are converted into binary vectors. Class one is presented as $(1,0,0)^T$, class two as $(0,1,0)^T$ and class three as $(0,0,1)^T$. The power-spectra used in the practical case of the eggs are 200-dimensional and are used without pre-processing to train the map. As for the spectra, the classification vectors are learned with the Kohonen learning rule. One may regard the input to the network as being split into two parts: one part that coverts the information about the spectra and one part that holds the information about the classes that the eggs belong to. In the training process, a cooling schedule is used for the Gaussian neighborhood function and for the learning rates. The second problem to deal with, is to make it possible to interpret several measurements from one object. The SOM algorithm is based on a distance measurement and this excludes the possibility to make an input vector that consists of the sequence of, in the case of the eggs, the six spectra and obtaining a 1200-dimensional vector. An example of this problem can be: consider two pairs of vectors that belong to two different objects: $(10,2,2)_1^T$, $(9,2,4)_1^T$ and $(8,2,4)_2^T$, $(10,2,2)_2^T$. With those two pairs, it is possible to make different combination vectors that have different distance, what proves the non-sense of combining vectors in one big vector. The answer to this problem is found by using a hierarchical SOM, which consists in fact of two different SOMs. The input of the second map is not a spectrum, but a vector containing the ordered activations of the first map which has been trained using each spectrum separately and which is completely unsupervised. If the dimension of the first map is i x i, the dimension of the activation vector is $(i^2,1)$. Initially, all values of this vector are zero. So, the number of elements (neurons) in the first map is the same as the number of elements in the activation vector. The reduction of the matrix that give the position of the neurons, which is denoted by A, to a vector V happens as follows: give each neuron, beginning with the left top corner of the map a number, ending with number i^2 on the right bottom corner of a i x i map and use this to fill in the activation vector V. Each time a neuron v has been activated, the numbers in the vector change:

$$V(v)^{new} = V(v)^{old} + 1$$

For each object, an activation vector will be formed. In this way, the activation vectors contain all information about the six spectra, because it covers all information about the winning neurons of the first map, which reflect perfectly the information of the six spectra. In the case of the eggs, a perfect egg will have six times a very similar spectrum, because it is radial symmetric around its equator. As a consequence of this, those spectra will hit six times the same neuron, leading to very characteristic activation vectors. Furthermore, because six measurements have been taken, the algorithm retrieves information about the seriousness of the breakage. The algorithm build so far, has one weak point that has to be eliminated. By taking the activation vector, the neighborhood of the neurons in the first map is lost. In the case of a 6 x 6 map, neuron (1,1) and (2,1) are neighbors, but are far away from each other in the vector. So, in this step, a very powerful tool of the SOM algorithm would be lost. A way of avoiding this problem, is to use a Gaussian activation instead of taking just the winner who produces a

one and all others a zero. The width of this activation results from experimental research. The optimum width that has been found is 0.3 x i. Because the map has not a fixed density all over its area, it is even better to adjust the width of the Gaussian accordingly to the density of the map. In this way, "local Gaussian activation functions" are formed. Mathematically, we can describe the hierarchical map as follows. A n-dimensional input **X** is presented to a i x i network. Euclidean distances d will be calculated for the whole map for finding the "winning neuron", namely that neuron whose weights ^{1} **W** (the index "1" denotes that the weight-matrix of the first SOM is meant) are closest to the input **X**. The neurons are named N_j (for j = 1:i²):

$$d_j = \left\| X - {}^1W_j \right\|$$

and for the winning neuron N_w:

$$d_w = \left\| X - W_w \right\| = \min_j \left\| X - W_j \right\|, \text{ for j = 1:I²}$$

Furthermore, we need an extra index to denote which observation of an object is meant. If there are m observations for one object, we use an extra superscript that represents the actual observation:

$$d_w^l = \left\| X^l - {}^1W_w \right\| = \min_j \left\| X^l - {}^1W_j \right\| , \qquad \text{for l = 1:m}$$

If we represent the horizontal dimension of the map by i_h and the vertical dimension by i_v, the distances are calculated between the actual activation center $\left(i_x^{w,l} \quad , \quad i_y^{w,l}\right)$ and the position of all neurons $\left(i_{x,j} \quad , \quad i_{y,j}\right)$ for j = 1:i². Consider the j-th element of the activation vector. Its value is calculated as follows:

$$a_j = \sum_{l=1}^{m} \left(e^{-\frac{\left(\sqrt{\left(\left(i_{x,j} - i_x^{w,l}\right)^2 + \left(i_{y,j} - i_y^{w,l}\right)^2\right)}\right)}{2\sigma^2}} \right)$$

Roughly, it can be said that the value of a_j is a function of the positions of the activation centers and the width of the Gaussian function:

$$a_j = f\left(\left(i_{x,l}^w, i_{y,l}^w\right) \quad , \quad \sigma\right)$$

In this way, the inputs $\mathbf{X}^l$ (l = 1:m) are combined in a vector which actually reflects the "path" followed by the inputs in the map. Because the positions of the neurons in the map are a dimension-reduction of the input information, **A** is supposed to carry all the information. Now, the activation vector **A** will be used as input to the second map to make it learn the "activation path" that was formed in the first map. As second input, the binary classes are fed to the network. The second SOM can be regarded as an ordinary SOM with the extension that it has an extra input to represent the class. The weights of this SOM are separated in two parts where the second part has c elements where c denotes the number of classes needed in the classification problem. The way these classes are learned is the same way as **A** is learned. The different steps explained above are summarized in figure 1. This figure has been designed for

only three measurements for one object (egg). The three neurons that are colored, represent the three "winners" for the three spectra (the black one is the actual winner, i.e. the winner for the third spectrum):

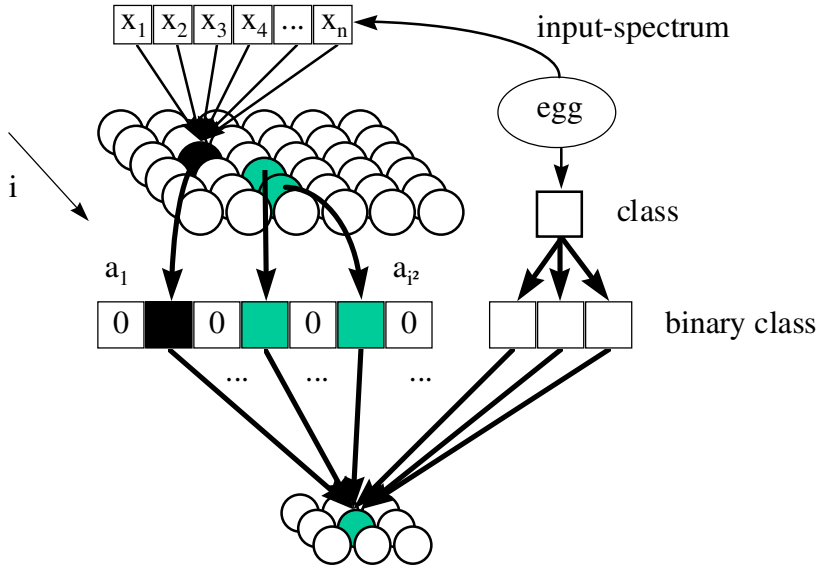

Figure 1: Training of the hierarchical SOM

After the net has been trained, a testing set will be presented to the hierarchical SOM. The output will be the class the object belongs to. This procedure is presented in figure 2:

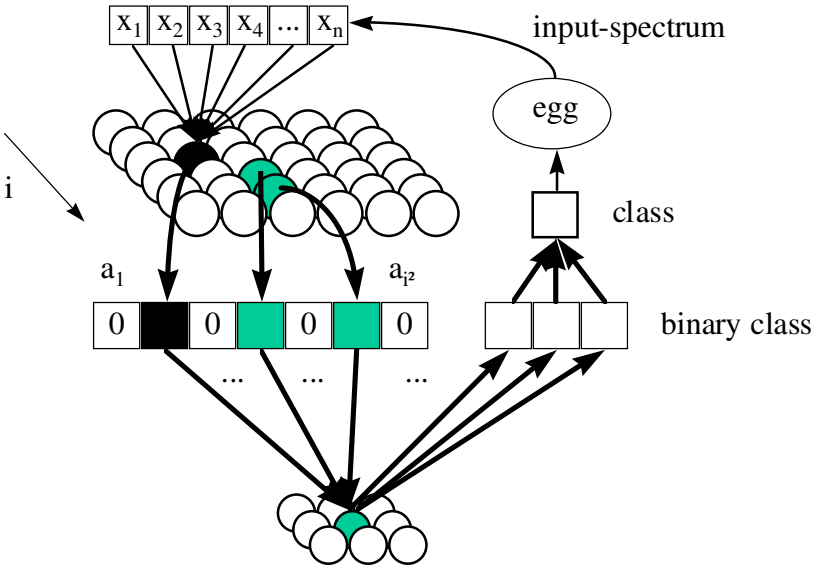

Figure 2:Testing of the hierarchical SOM

The only thing that changes between the training and testing procedure, is the input to the second map. Only the activation vector **A** is given as input now. Afterwards, the Euclidean distances are calculated as follows:

$$d_r = \left\| A - {}^2W_r(q) \right\| \ , \ \text{for } r = 1{:}i^2,\ q = 1{:}i^2$$

Remark that the distance is, of course, based on the i^2 elements of ${}^2\mathbf{W}$ only and the superscript "2" is used because the weights of the second SOM are meant. For simplicity it has been assumed that both maps have i^2 elements. The output of this network will be:

$$a^w = {}^2W^w\left(\left(i^2+1\right):\left(i^2+c\right)\right)$$

So, a^w represents the classes learned by the network. This values are not binary but continuous between 0 and 1. The more these values are closer to 1 or 0, the better the classification problem is solved by the network.

III. Experimental results

Some experimental results will be given for the egg-application. Each egg is hit six times by a small impact hammer around its equator. These signals are collected and fed to a spectrum-analyzer which converts the time signals into power-spectra. These power spectra form the input of the hierarchical SOM. The best results are obtained with the Gaussian activations: for the class 1 eggs, 97.5%. For class 2, 84% was classified correctly and 65% of class 3. Without the Gaussian activations, only The lower percentage of the class three classification is caused by its definition. Even a very small crack where the membrane is broken, must be classified theoretically as class three. Without the Gaussian activations, the percentage of correct classification was lower, especially for class one (only 83.5 percent correctly classified). However, mechanically, class one is closer to class two than class three. Maybe a re-definition of the classes is needed in order to obtain better classification. They need to be classified according to a "probability of becoming a class three during the processing". This classification problem has been approached also with second-order statistical methods and heuristic algorithms in our Lab, but none of them was able to compete with the hierarchical Self-Organizing Map.

IV. Conclusion

A very powerful classification method based on the Self-Organizing Map has been presented above. The method can deal with all classification problems where several measurements have been taken from one object. In other words, this method can be applied in a large range of applications.

References

H. Ichiki, M. Hagiwara, M. Nakagawa, (1993) "Kohonen Feature Maps as a Supervised Learning Machine", *Proceedings of the IJCNN-IEEE 1993 Conference*, S. Francisco CA, pp. 1944-1948

T. Kohonen, (1982) "Self-Organized Formation of Topologically Correct Feature Maps", *Biological Cybernetics*, 43, pp. 59-69

T. Kohonen, (1995) *Self-Organizing Maps* , Springer Series in Information Sciences

H. Ritter, T. Martinetz, K. Schulten (1992), *Neural Computation and Self-Organizing Maps,* Addison Wesley Publishing Company