

Entropic Optimum Synthesis of Multi-layered Feed-forward ANNs

Andrea Pelagotti, Vincenzo Piuri

Department of Electronics and Information, Politecnico di Milano
piazza Leonardo da Vinci 32, I-20133 Milano, Italy
piuri, pelagott@elet.polimi.it

ABSTRACT

Optimization of the neural architecture is often critical to design an efficient and feasible solution, in particular when a VLSI implementation is considered. This paper proposes an original approach to the synthesis of multi-layered feed-forward ANNs based on the analysis of the information quantity flowing through the network. A layer is described as an information filter which selects the relevant characteristics until the complete classification is performed. The basic incremental method, including the training supervised procedure, is derived to design optimum (or nearly-optimum) neural paradigms. A significant variant is also proposed to improve performances.

1. Introduction

Design and tailoring of a multi-layered feed-forward neural network to solve a specific application problem (e.g., pattern classification) implies identification both of the topological network structure (i.e., the number of layers and their sizes) and of the operating parameter configuration (i.e., interconnection weights and thresholds). The traditional approach consists of creating a tentative network by exploiting the knowledge of experts, training the uncommitted structure by using the back-propagation algorithm or similar ones, and validating the final configured network to verify the operation error and the generalization capabilities. Unfortunately, this approach has two drawbacks: the topological structure must be defined a priori before starting the learning phase, and the resulting structure may be overdimensioned with respect to the specific application. Pre-definition of the structure is strictly related to the capabilities and the expertise of the designer: if the network is not large enough, the training procedure may not be able to converge to a zero learning error. A too large network may not be able to generalize - in a satisfactory way - the learnt behavior in the presence of unknown data presented during validation or recall.

Besides, the network size may be a critical factor for physical realizability when VLSI/WSI circuits are adopted for emulating the neural operations. Great difficulties in data path design and in the evaluation of the neuron's activation signal may in fact occur when the interconnections are too many. Similarly, the computational complexity may require a powerful traditional computer when the network operation is simulated by software and real-time constraints are imposed by the application. To solve these problems, the traditional strategy overdimensions the neural paradigm for initial training and, then, reduces its size by means of optimization algorithms (e.g., [1, 2]): a

new training phase may be envisioned to adapt the configuration parameters of the reduced structure to the given application. Some learning procedures (e.g., [3]) provide a partial support to network optimization by incorporating minimization goals into the nominal algorithm.

In this paper, we propose a new synthesis method which builds the network structure incrementally during the learning phase. Our approach is based on the analysis of the information quantity (the *entropy*) flowing through the network layers. In Section 2, each layer is viewed as a filter which compacts the information transferred from the layer input to the output, according to the classification required by the application problem. The multi-layered feed-forward network is created one layer at a time and one neuron at a time during the learning phase, by adding layers and neurons according to the topological modifications which lead to control the information flow as expected by the application. The resulting network has the minimum topology (i.e., number of neurons) which is necessary to solve the application and, as a consequence, the maximum generalization capability. The neural computational capabilities are equivalent to the ones of a network created by using back-propagation. The new learning-optimizing procedure is presented in Section 3. On the contrary of the back-propagation algorithm (or similar ones) convergence of our technique to a working network is always guaranteed since such a network is obtained by incremental construction and not by simple parameter tailoring of a given structure.

In some cases, extensive experiments have shown that our approach is not able to identify the globally-minimum structure, but only a nearly-optimum solution requiring very few (usually, less than three) neurons more than the optimum one. In Section 4, a variant of our algorithm is presented to deal with several of these cases.

2. Neural layers as information filters

According to C. E. Shannon [4], the quantity of information (H) generated by an information source is measured by $H = -k \sum_{i=1..n} p_i \log p_i$, where n is the number of different messages produced by the source and p_i is the probability of i -th message. If we set $K=1$ and select the logarithm base equal to 2, H is expressed in "bit/msg". Since previous equation is similar to the statistical thermodynamics entropy, the information quantity H is called *entropy*.

Every layer in a multi-layered feed-forward network receives information from the previous level and sends information to the next one. In every layer, the information is represented by the vector of values that are generated at the outputs of neurons belonging to the previous layer and, as a consequence, that are presented at the layer's inputs. The vector of input values is called the *message* entering the layer.

A pattern presented at the network's inputs is transmitted by the input layer to the first hidden layer without any modification; the first hidden layer computes the resulting message and delivers it to the next layer, and so on until the output layer generates the network's output message. If the step function is used in neurons, all messages are bit strings (possibly, except the messages in the input layer, because the input patterns of the specific application problem may be real numbers).

For each layer, we compute the input and output quantities of information H , by applying all the input patterns at the network's input and measuring the relative frequencies p_i of the different messages which are received and transmitted by the layer. The number of messages produced by a layer is not higher than the number of received messages. Each input message is obviously mapped onto at most one output message; two or more input messages may be collapsed by the application into only one layer's output message (in such a case, the individual probabilities p_i related to these merged input messages are replaced by their sum in the set of layer's output messages). Therefore, it is $H_{in} = \sum_i (p_i \log p_i) \geq (\sum_i p_i) \log (\sum_i p_i) = H_{out}$. The equal sign holds if the number of messages is not changed.

Within each layer, we introduce the specific entropy h related to the average quantity of information associated to every neuron of such a layer. It is defined as the ratio between the layer entropy H and the number of neurons in the layer. Indirectly, for a given topology, it gives a measure of the information concentration and the relevance of that number of neurons for the solution of the application problem with respect to other topologies.

The input patterns and the overall information flowing through a multi-layered neural network contain two components: one is necessary to solve the application problem correctly, the other is redundant. The first

one must be preserved through the layers so that the correct classification of the input pattern occur, while the second one should be removed by the layers as much as possible to extract the true nature of the application problem and, as a consequence, to grant high generalization capabilities. The individual layer of a neural network can therefore be viewed as an information filter which tries to remove as much redundant information as possible, without losing useful information. Both the above components contribute to the total entropy H_{out} at layer's output.

Considering the individual hidden layer, experiments have shown that the output entropy H is minimum when the layer size has the minimum value required to solve the application, while the specific entropy h is maximum around the minimum size. In Fig. 1, the classic case of detecting the symmetry in 4-bit patterns is shown.

To analyze the information flow, it is necessary to measure the loss Δ of useful information. Let's consider the first hidden layer of a multi-layered network for a typical classification application. The network's input patterns belonging to c different application classes are applied one at a time at the network's inputs: c classes of layer's output messages are generated according to the application specifications. Each class of input patterns is transformed by the layer into a class of output messages. The *partial classification* made by the layer is said to be *correct* if any pair of output classes is disjointed. If an output message belongs to more than one output set (i.e., there is at least one pattern in more than one input class that is mapped onto the same output class), the next layer cannot distinguish some input patterns belonging to different input classes so that they will be erroneously grouped in the same class. Messages belonging to more output sets are called *ambiguous*. Ambiguous messages induce losses in the quantity of useful information. The loss Δ_I of useful information in the first hidden layer can be evaluated by subtracting the real entropy H_{out} from the entropy H_{out}^* computed by assuming that no message is ambiguous. We obtain $\Delta_I = \sum_{i=1..c} H_i^I - H^I$, where H_i^I is the entropic contribution due to i -th class of input pattern. The entropies are measured at the outputs of the first hidden layer. Every p_i is the ratio between the number of occurrences of the i -th message and the number of input patterns. Similarly, we can measure Δ_j for the j -th hidden layer ($j > 1$) by applying to the network input all patterns and measuring H^j and H_j^j at the outputs of the considered layer. The above definition leads to compute Δ_j as the measure of the useful information lost by the j -th layer and by its predecessors. If the partial classification of its predecessors is correct, Δ_j measures the useful information lost only by the j -th layer.

3. The optimum synthesis technique: the Δ method

Traditional techniques for topological optimization are concerned with minimization of the layers' sizes and interconnections, by operating on a completely-trained network or by introducing a penalty factor in the learning procedure to remove neurons or interconnections. Usually, the initial topology is selected by the designer based on his expertise without any specific *a priori* rule supporting his choices.

In this paper, the synthesis approach for multi-layered feed-forward networks (having the step function as non-linear activation function) is reverted and based on direct creation of the optimum (i.e., topologically-minimum) network directly during learning. One layer at a time and one neuron at a time are built, by adding layers and neurons according to the topological modifications which lead to control the information flow as expected by the application.

The theoretical foundation of our technique is constituted by the following theorem: *For each layer j , the loss Δ_j of useful information is zero and the number of classified messages at network output is c if and only if the network classifies correctly.* The first condition ensures that every layer does not lose any useful information, i.e., the output layer produces different messages when, at network input, we apply patterns belonging to different classes. The second condition states that the number of network output messages is equal to the number of the required classes. The network classifies correctly when the learning error is zero. If the two conditions hold, the network is able to distinguish all the input classes exactly since no classification information is lost through the layers and the number of classes is equal to the expected value. Conversely, if the classification error is zero, it is obvious that all patterns are correctly recognized, i.e., no relevant information has been lost by filtering layers and all expected classes are present; in fact, if at least of such conditions was not be true, the resulting learning error should be non zero. This theorem can be exploited to build an optimal multi-layered feed-forward network by searching for the smallest neural structure which respects both the above conditions. Training on the input patterns is used to select the "best" solution at each step of the algorithm. The new synthesis technique (here the Δ method) is summarized as:

1. *Initialize the network topology:* The neural structure is initially composed of the input layer only.
2. *Create a new layer:* A new layer is added to the network topology by using the minimum number of neurons which is necessary to transmit all the useful information received by the previous layer (i.e. $\Delta=0$). Operations are as follows:

a) *Initialize the layer topology:* The new layer is set to an empty layer.

b) *Create a new neuron:* A new neuron is added to the layer under construction.

c) *Set the new weights:* The interconnection weights for the new neuron are evaluated so that Δ for the layer under construction is minimum. To measure Δ , the output messages of the previous layer are applied to the layer under construction; H and H_i are computed at the output of such a layer. Only the neurons introduced till that time are considered to compute the above figures of merit. A local optimization procedure should be adopted to evaluate the best weight set; to increase the computation speed (without affecting the overall optimality of the final result) the weight set inducing the minimum Δ can be selected among a number (large enough) of weight sets randomly generated.

d) *Layer completion:* If $\Delta=0$, the layer synthesis ends; otherwise, it continues from step b.

3. *Network completion (convergence check):* creation of new layers terminates as soon as both the condition of the above theorem are satisfied, i.e., the network is able to perform the required classification, correctly. The number of messages is counted at the network output: if it is equal to the number of required classes, then the synthesis ends, otherwise it continues from step 2.

Note that only one neuron at a time is manipulated, so that the number of variables (the interconnection weights) is smaller than in other learning algorithms (e.g., back-propagation). In our experiments, we chose the best weights in a set of randomly-generated configurations: in particular, we generated 10^4 random positions for the hyperplane corresponding to the set of weights associated to the new neuron.

In Fig. 2, the application of this technique is shown for the simple classification problem with two discrete-value input variables: patterns are classified in two classes identified by black dots and empty circles, respectively. The hyperplanes generated by the algorithm are progressively numbered as soon as they are created. Our synthesis method converges for any classification problem, i.e., always terminates by generating a network capable of solving the given application problem. In fact, a finite number of neurons is always needed in any layer to obtain that Δ is equal to zero (*layer convergence*). This is a direct consequence of the generally-monotonous decreasing behavior of Δ . When a new neuron is inserted in the layer under construction, its weights can be always set to minimize Δ , which becomes smaller or equal than the value before neuron insertion. It can be shown [5] that Δ does not decrease only when the following (infrequent) condition occurs and that only a limited number of occurrences of such a condition are

allowed. In the input hyperspace, a convex region which contains patterns belonging to more classes (see Fig. 3a) is called *ambiguos* since the output message generated by applying each of such patterns is ambiguos. Δ does not decrease (Fig. 3b) if and only if neuron insertion introduces a hyperplane that does not change the ratio between the number of patterns belonging to different classes. This situation cannot persist indefinitely: in a finite number of steps (i.e., neurons), Δ is reduced to zero (i.e., there are no more ambiguos region).

On the other hand, a finite number of layers is always needed to create a neural network for any classification problem (*network convergence*). Consider first the parity bit problem. This is the most difficult classification problem: in fact, the input patterns are placed on the vertexes of an n -dimension hypercube and any pair of patterns in the same class are not adjacent. It can be easily shown that this problem is successfully solved by our technique; experiments have proven that one hidden layer (composed of as many neurons as the number of input bits) is sufficient. Since one hidden layer is able to solve the most complex problem (i.e., the parity bit) among the binary-pattern classification problems, it is obviously able to solve also any other (more simple) case of such problems. Two hidden layers are required in the worst situation to classify non-binary input patterns: the first layer is used to binarize the input patterns, while the second one performs the binary classification.

Extensive experiments have shown that our method is equivalent to traditional training approaches (e.g., the back-propagation algorithm) as the learning error and the generalization ability are concerned. In traditional methods, these figures must be evaluated on networks minimized by applying optimization techniques (e.g., see Section 1).

Extensive experiments have shown that our synthesis technique is generally able to create the minimum network (i.e., with the minimum number of neurons within each layer). In some cases, the absolute minimum value is not reached, but a quasi-minimum solution is always achieved. In fact, minimizing Δ after inserting each neuron is not exactly equal to a more global minimization of Δ after two or more neuron insertions. Anyway, extensive experiments on a wide class of classification problems have shown that redundancy is limited to no more than three neurons with respect to the absolutely minimum solution.

4. A variant of the synthesis method: the Δ/H method

As remarked in the previous section, the Δ synthesis method based on the pure minimization of the entropy H is not always able to guarantee creation of the

globally-minimum network. An evident example (for which it is even trivial to create the input space partitioning by hand at a first glance) is shown in Fig. 4. Two classes must be recognized: the first one is composed by the five square groups of black dots, and the second one by the remaining empty circles. Fig. 4a gives the hyperplane tessellation of the input space provided by first hidden layer of the four-layer network with 2-11-4-1 neurons, created by the Δ algorithm; Fig. 4b shows an alternative intuitive solution for the first hidden layer, which requires a smaller number of hyperplanes (layers contain 2, 8, 2, 1 neurons, respectively). Non-minimality of the first network can be directly observed in the behavior of the functions H and Δ evaluated at each step during the algorithm evolution, as shown in Fig. 5a and Fig. 5b, respectively. In the network of Fig. 4b, H reaches the maximum value and Δ goes to zero after less neurons have been inserted in the hidden layer than in the network of Fig. 4a. This is an experimental proof that minimizing Δ locally for each new neuron may not lead to the globally minimum Δ value that can be obtained by minimizing Δ for all the units: in Fig. 5b, after inserting 4 neurons, the Δ value of the second network drops below the first one.

Enhancement of our synthesis method may be achieved by adopting a lightly-different entropy-based function that must be minimized at each neuron insertion during layer creation. First of all, minimization of such a function must imply minimization of the Δ function and, in particular, Δ must be zero when minimization ends. Then, the new function must be less sensitive to the possible presence of partial minima than the Δ function.

Consider the entropy H measured at the output of the first hidden layer of network, as it is shown in Fig. 5a. Note that its value for the network of Fig. 4b is always higher than for the network of Fig. 4a. Unformally, we can remark that the synthesis method should try not only to reduce Δ to zero, but also to maximize H in order to escape from the locally-minimum final structure and to try to reach the globally-minimum one. In other words, we should try to avoid to loose relevant information (i.e., push Δ to zero), but - at the same time - we should maximize the quantity H of information transmitted to the next layer. To obtain this effect, we extensively experimented the use of the function Δ/H in the minimization procedure instead of the function Δ . The new resulting approach is called the Δ/H method: operations are performed as in the Δ method discussed in Section 3, but Δ is replaced by Δ/H .

A side effect of this modified technique is related to the placement of hyperplanes in order to obtain rather large convex regions. Sacrificing a further Δ reduction, we are able to preserve larger regions and, as a consequence, to achieve better results in the

subsequent classification steps. In Fig. 6, we show two possible placements of the first hyperplane in the hidden layer of the example of Fig. 4; the solution proposed in Fig. 6a implies a modest reduction of Δ , but a better ratio Δ/H and an optimal hyperplane placement for the subsequent iterations. As example, in Fig. 7, we show the final tassellation of the input space that is induced by the hidden layer created by the Δ/H method. For comparison, we give the functions H and Δ for the two synthesis methods in Figg. 8a and 8b, respectively.

Even if the previous result is relevant, however, extensive experiments have shown that the Δ/H method cannot be considered as the general solution for the optimized synthesis of multi-layered feed-forward networks. In fact it is not always more effective than the simple Δ method. In some application problems, we observed that the Δ/H method produces networks larger than the Δ method. An example of this misbehavior is the classification problem presented in Fig. 9: the first class is composed by black circularly-placed dots, the second one by the empty circles around the black dots. In Fig. 10, we give the functions H and Δ for the two synthesis methods. Maximization of H in the Δ/H method has allowed more information to pass through the hidden layer than in the Δ method, so that a lot of useless information is transmitted to the next layer and more neurons are required to obtain a zero Δ .

Since no rule has been found yet to identify a priori when the Δ method is more suited than the Δ/H method, both synthesis methods can be applied and the smallest network is chosen. This allows for exploiting directly the best performance of our methods. An alternative approach consists of applying only one of the two synthesis methods: the resulting network is then minimized by using a traditional optimization technique. Even if this solution is very similar to the traditional approach described in Section 1, there are two relevant advantages. First, no large overdimensioning of the neural paradigm is required for the initial training since the synthesis method generates an optimum or quasi-optimum network, so that minimization will be easier. Second, the synthesis method always creates a network with zero learning error, while traditional learning algorithms are sometimes not able to configure the weights correctly even if the architecture has enough neurons due to the presence of local minima.

References

- [1] C. Alippi, V. Piuri, "Topological Minimisation of Multi-layered Feed-forward Neural Networks by Spectral Decomposition", *Proc. IJCNN'92*, Beijing, China, November 1992
- [2] S. Tamura, M. Tateishi, M. Matumoto, S. Akita, "Determination of the Number of Redundant Hidden

Units in a Three-layered Feed-forward Neural Network", *Proc. IJCNN'93*, Nagoya, Japan, 1993

- [3] J. Hertz, A. Krogh, R. G. Palmer, *Introduction to the theory of neural computation*, Addison-Wesley, 1991
- [4] C. E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, vol XXVI, n. 3, July 1948
- [5] A. Pelagotti, V. Piuri, "Entropic analysis of feed-forward multi-layered neural networks", Politecnico di Milano, Department of Electronics and Information, Report No. 034-95, 1995

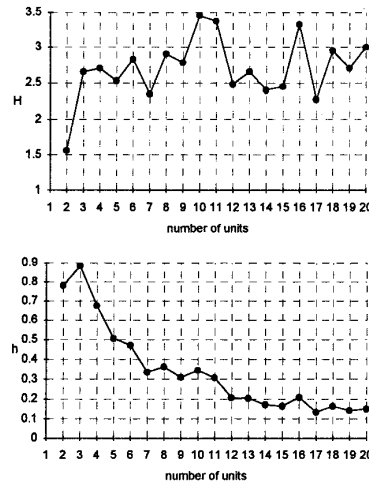


Fig. 1 - The entropy H and the specific entropy h for the symmetry detection problem in 4-bit patterns.

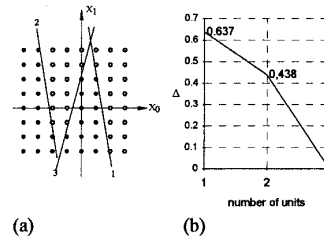


Fig. 2 - Generation of minimum first hidden layer: input tessellation (a), function Δ (b).

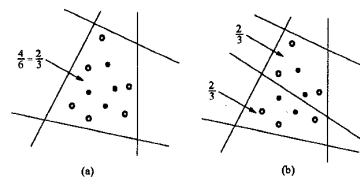


Fig. 3 - An ambiguous region before (a) and after (b) neuron insertion with unchanged Δ .

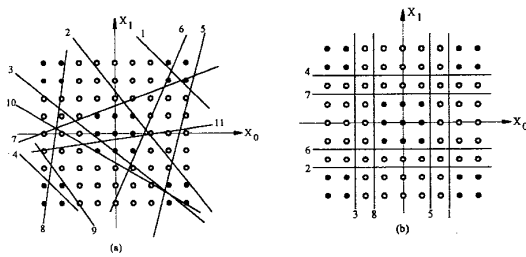


Fig. 4 - Difficulties for the Δ method in solving a classification problem.

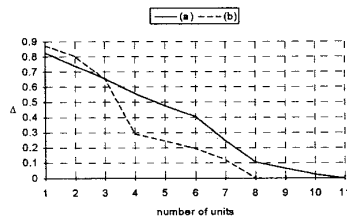
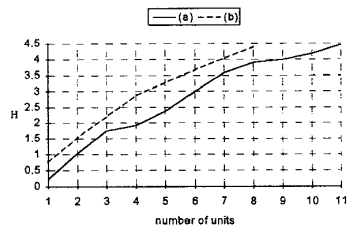


Fig. 5 - H and Δ of the first hidden layer in the networks of Fig. 4.

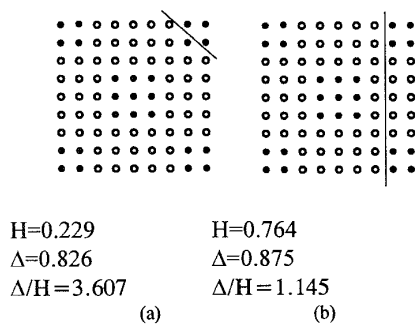


Fig. 6 - Alternative placements for the first hyperplane.

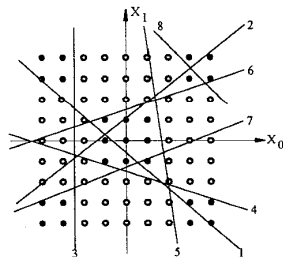


Fig. 7 - Hyperplanes placement in the first hidden layer for Δ/H synthesis.

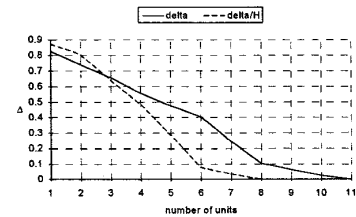
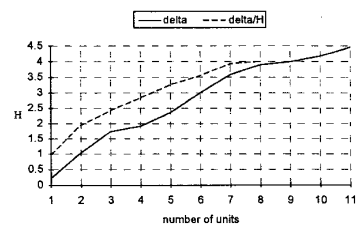


Fig. 8 - H and Δ in networks synthesized by using Δ and Δ/H methods for the problem of Fig. 4.

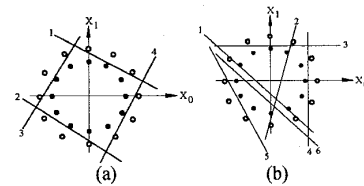


Fig. 9 - Another classification problem: the use of the Δ method (a) and Δ/H method (b).

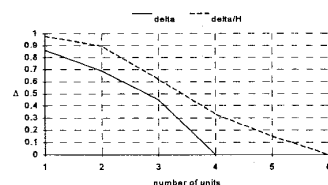
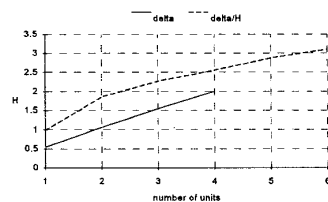


Fig. 10 - H and Δ in networks synthesized by using Δ and Δ/H methods for the problem of Fig. 9.