International Journal on Artificial Intelligence Tools (IJAIT)

COUPLING SUPERVISED AND UNSUPERVISED TECHNIQUES IN TRAINING FEED-FORWARD NETS*

CRIS KOUTSOUGERAS and GEORGE PAPADOURAKIS†

Computer Science Department, Tulane University, New Orleans, LA 70118, USA

Department of Computer Science†, University of Crete, Heraklion, Crete, Greece

ABSTRACT

A popular approach to training feed-forward nets is to treat the problem of adaptation as a function approximation and to use curve fitting techniques. We discuss here the problems which the use of pure curve fitting techniques entail for the generalization capability and robustness of the net. These problems are in general inherently associated with the use of pure supervised learning techniques. We argue that a better approach to the training of feed-forward nets is to use adaptive techniques that combine properties of both supervised and unsupervised learning. A new formulation of the training problem is presented here. According to this formulation the net is viewed as two coupled sub-nets the first of which is trained by an unsupervised learning technique and the second by a supervised one. The same formulation gives rise to analytic expressions of the goals of the adaptation and leads to a new method for the adaptation of feed-forward nets.

Keywords: neural nets, curve fitting, learning, function approximation.

1. Introduction

One of the primary targets of artificial neural networks research is the development of computational models for approaching the problem of learning by examples. This problem can be stated as the automatic identification of some function where

[0]*This research is supported by NSF grant MIP-8910616 and by Tulane's COR

all that is known about the function is a finite set of input-output sample pairs. Thus the problem entails the automatic development of hypotheses concerning the identity of the function from which the samples come from. The only solid criterion for evaluating such a hypothesis is whether the corresponding plausible function is consistent (or explains, or fits) the sample set [1]. Thus a popular approach to the problem is to treat it as a function approximation one and to use curve fitting techniques for the adaptation (or training) of the net. However, focusing on fitting the training set per se, like with most adaptive techniques of purely supervised nature, leaves the development of the net's internal representations up to random parameters and factors of the training process as long as a variety of different representations can result in proper fitting. This is no more than a statement of the fact that there are many different functions which can fit a given sample set and one is selected randomly among them if selection is based only on the criterion of proper fitting. However, this selection corresponds to drawing a generalization from the sample set and this generalization is supposed to capture the "semantic" properties of the sample set. The generalization is required to be such that not only the sample set can be accurately reconstructed, but also to provide the means for a reasonable interpolation or prediction for other input-output pairs of the (original) sampled function (pairs that were not in the given sample set). So the random character of the function selection associated with pure supervised learning techniques severes the reliability and robustness of the adaptation. In this paper we discuss the problem of generalization and we conjecture that an additional criterion should be used for evaluating the function selection: of all the functions that fit the sample set we seek the one of least nonlinearity. The requirement about reduced nonlinearity relates to the net's structure and the net's internal representations which develop as a result of the adaptation.

As mentioned earlier, the use of pure supervised learning techniques leaves the development of internal representations potentially prone to random factors such as available nonlinearity, initial parameter values, etc. The function of a net can be viewed as a two stage process and we suggest here that there are advantages in using adaptive techniques which treat it as such. Based on this view we suggest that the adaptation of a feed-forward net can proceed in two stages and that these two stages do not necessarily need to be handled by a single *uniform* process or technique. Unsupervised Learning techniques can be employed for one stage of the adaptation while supervised learning techniques can be employed for the other stage. By introducing elements of unsupervised learning in training techniques of supervised nature, better guidance and control is exercised on the formation of the internal representations.

Following the discussion regarding the issue of generalization, we present the view that the function of a general feed-forward neural net can be considered as a recoding followed by curve fitting and that unsupervised learning techniques can be employed to aid the recoding function. We then introduce a new formulation of the training problem for feed-forward nets. With this formulation the goals of the adaptation can be expressed in terms of analytic expressions, and also, bounds on the optimal net structure for proper adaptation can be concluded. Finally, this formulation leads to an outline of a class of adaptive algorithms for feed-forward nets.

2. Background

In this paper we focus on feed-forward nets. In feed forward models the network structure is such that the output of each neuron does not affect itself, that is, the output of each neuron does not contribute, either directly or transitively, to its own input. For such networks without feedback, the topology reflects a dependency graph. Neurons are arranged in layers so that the total input stimulus to each neuron depends only on neurons in previous layers. The output of the net is a vector consisting of the output values of the neurons in the output layer. If the inputs to the net are vectors from an n-dimensional space V and there are k neurons in the output layer then the (transfer) function F of the net is a mapping from V to R^k. Since there are no lateral interactions among neurons in any given layer, F can be assumed decomposable into k functions $F_i : V \rightarrow R$, for i=1,...,k. Thus, without loss of generality, we can restrict our discussion to nets with a single output. All observations then pertain to each of the F_i's in the case of many output neurons.

Consider now the simple net shown in Fig. 1. This is a typical sub-structure of those which compose a general feed-forward net. Let F be the output of the net in Fig. 1, and let O_i and f_i be respectively the output and the activation or squashing function of neuron n_i. Let also W_{ij} be the connection strength or weight from n_i to n_j. The total net output being the output of neuron n_5 can be expressed as :

$$\begin{aligned} F &= O_5 = f_5(net.input_5) = f_5(W_{45}O_4, W_{35}O_3, W_{25}O_2) \\ &= f_5(W_{45}f_4(W_{14}O_1, W_{34}O_3), W_{35}f_3(W_{13}O_1, W_{23}O_2), W_{25}O_2) \\ &= f_5(W_{45}f_4(W_{14}f_1(inp_1), W_{34}f_3(W_{13}f_1(inp_1), W_{23}f_2(inp_2))), \\ &\quad W_{35}f_3(W_{13}f_1(inp_1), W_{23}f_2(inp_2)), W_{25}f_2(inp_2)). \end{aligned} \tag{1}$$

The above equation shows that with such a feed-forward structure the net's function

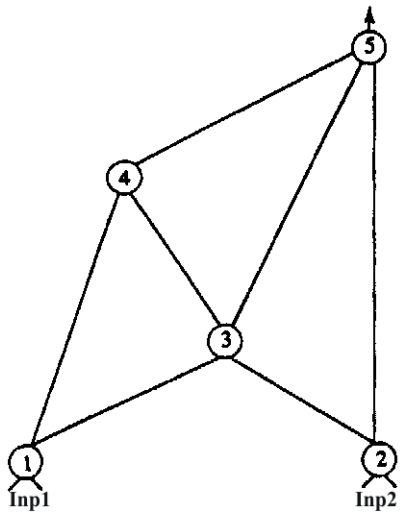

Figure 1. Example of a feed-forward structure.

is a composition of the simple activation functions of the net's neurons. The transfer function of any arbitrary feed-forward net can be expressed in a similar way if we start with the expression of the function of an output neuron and recursively substitute corresponding expressions for its inputs. The final expression will be an analytic form expressing the net output as a parametric function of the net's inputs where the connection weights are the parameters.

In the following, we will refer to the specific sub-functions which synthesize the analytic form of the net's transfer function as *descriptors* or *higher order features*. For example, in Eq. (1), $f_2(inp_2)$, and $f_3(W_{13}f(inp_1), W_{23}f_2(inp_2))$, and $f_4(W_{14}f(inp_1), W_{34}f_3(W_{13}f(inp_1), W_{23}f(inp_2))$, etc., are descriptors for F. Note that there is a distinction between an activation (squashing) function and a descriptor; a descriptor is the output of a neuron as a function of the network's input (as opposed to the neuron's inputs).

The analytic expression for the net's transfer function, like that of Eq. (1), rapidly grows more complex as the number of neurons and layers increases. The number of layers defines the depth to which the activation functions are embedded. A point to be noticed here is that even with minor expansions to the net structure, the nonlinearity of its total transfer function can easily be increased to the levels required for fitting arbitrary sample sets or for the approximation of a large variety of target functions.

Usually, the form of the activation functions f_i is a very simple one and it is fixed (all neurons have the same activation function), so the net's transfer function is in practice a parametric function of the net's inputs where the parameters are the connection strengths or weights. These weights are flexible parameters which are determined during the net's training phase. Thus the adjustment of the weights is usually achieved with algorithms based on parametric function approximation. The set of target functions which in principle can be approximated within any desired accuracy depends on the nonlinearity available through the net's transfer function, e.g. Eq. (1), which in turn depends on the particular form and number of the f_i's and the way they are embedded in the analytic expression of the transfer function.

The goodness of fit of F to the sample set (or the goodness of the approximation) is measured by some (positive definite) function E_s (such as the total output square error). The fact that F is analytic permits the determination of how changes in each specific weight affect changes in E_s. The adaptive mechanism thus induces stepwise changes in the weights which cause E_s to decrease to a minimum value. Thus this mechanism simulates a gradient descent according to the rule: $\Delta W = -\alpha \frac{\partial E_s}{\partial W}$ known as the generalized Delta rule [2]. The adaptive process terminates when E_s cannot be reduced further. There is of course the question of this gradient descent getting stuck in local minima of E_s although quite effective techniques, such as simulated annealing, have been in use [3, 4]. However, the most difficult problem is to determine the net's topology – which in most models is chosen a-priori. One must virtually guess the necessary number of layers and neurons per layer which would facilitate proper approximation of a specific target function F_T by the net's function F. The adaptation mechanism operates on a net of predefined topology and it is not easy to determine whether the nonlinearity of a given net is sufficient for a certain target function F_T. The problem with the number of layers may not be serious because it has been shown that two layers (one hidden) are sufficient for a wide variety of target functions (case of sigmoid f_i's) [5]. In addition, determining the number of neurons per layer remains a difficult open issue.

The choice of topology (structure) which would be suitable for approximating a given F_T often requires experimentation with various nets of different topologies. This is hardly a practical solution though. Besides, it does not provide a reliable topology choice either. Assume that with a given net, the fitness E_s cannot decrease down to 0. Then it is obvious that the net does not contain enough nonlinearity. On the other hand, assume that a 0 for E_s can be attained. This means that the net's function F coincides with the target function F_T at least at the points of the sample set, but there is no guarantee that F_T is approximated properly in other points. In other words, a 0 for E_s may imply the possibility that the net contains

more nonlinearity than is exactly necessary.

3. The Issue of Nonlinearity

The presence of excessive nonlinearity may induce an unfavorable net behavior (sometimes random) outside the specific instances of the training set. In general, given a set of points in some n-dimensional space (training set or set of samples) there exist many different surfaces (functions) S_i, i=1,..., which fit those points. This is illustrated in Fig. 2. A curve fitting approach which is based only on the objective of discovering a surface which contains the sample points, such as the Back Propagation Algorithm (BPA) [2], will at best select *randomly* one of the above candidate S_i's. So if there is excessive nonlinearity, then it is possible to represent many different functions, all of which fit exactly the sample set. One of these is randomly identified by the adaptive mechanism since this identification is heavily dependent on various parameters of the adaptive mechanism such as initial weight values, step size (reflected by the learning rate *alpha*), momentum terms, etc. Reports [6, 7, 8] exist which specifically refer to the sensitivity of Back Propagation nets to the choice of the initial weights. This random character of the end result of the adaptation is quite likely to result in unpredictable or unreliable behavior of the net after training. The formulating of the learning problem as a curve fitting one is appropriate but also too general. The curve fitting problem, in the absence of additional constraints, does not in general admit a single solution and thus it is bound to have a random character. This is severe for the reliability of current adaptive techniques and subsequently for the usability of the models under consideration. What is really expected of the adaptation is not only an issue of the goodness of fit but also an issue of the quality of the developed internal representations (i.e. has the net simply "memorized" the training set or has it really captured its "semantic" properties?).

What is evident from this discussion is that focusing only on fitting the sample set, as it is usually done with purely supervised learning techniques, does not touch upon the issue of what kind of *generalization* the net develops as a result of the adaptation. What is considered a "reasonable" generalization though? Given the set of functions all of which fit a sample set, which one is it reasonable to assume as being the sampled one? Since the original function (the one which was actually sampled and which is also the target of the adaptation) is not known there really isn't any solid criterion other than the consistency to the sample set. Since all that is known about that function is the sample set, any function consistent with the

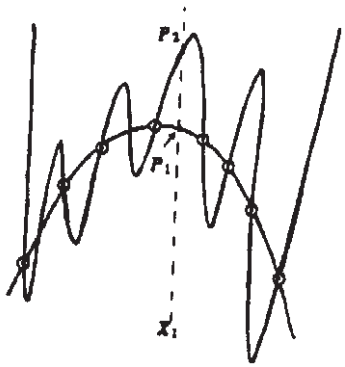

Figure 2a. Two ways to fit a sample set

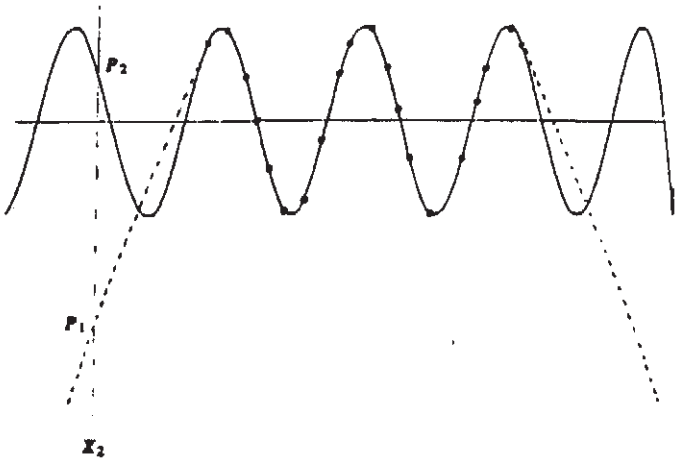

Figure 2b. Two ways to fit another sample set

sample set is a legitimate candidate. However, in many cases it is assumed that the target function has been sampled densely enough and that the sample set points out the basic form or shape of that function reasonably well. Consider the example of Fig. 2a. Any of the two curves shown could be the one which the samples came from. We then ask what is the output which expectedly corresponds to the input marked with "X_1"? There are two outputs as suggested by the two possible functions/generalizations shown. Based solely on the reflection of the sample set *alone*, which of the two outputs would be most likely anticipated? One of the choices seems to follow a reasonably evident pattern (shape in this case) whereas the other does not. Recall that we assume the target function to have been sampled densely enough as to capture recurring patterns or "semantic" properties.

Even if in the above discussion we relax the assumption that the target function is densely sampled, we are still left with the fact that any one of the functions which are consistent with the sample set is as good generalization as is any other one of them and beyond the criterion of consistency the selection of the generalization hypothesis is random. So there is no loss in introducing further selection criteria even if arbitrary. We conjecture here that such a reasonable criterion to add is the requirement of reduced nonlinearity. In other words, we consider our target to be the function of least-nonlinearity which fits the sample set. There is one more point to clarify with regard to this requirement. Consider the example of Fig. 2b. Suppose now that we ask what is the expected output for the input designated as "X_2" (the same question asked earlier about the input "X_1"). In this case, the requirement for a fitting function of least nonlinearity would suggest that the point P_1 is more likely than P_2. Given our intuition about the graphic pattern of the plot of the sample set (which looks like a sinusoidal), we might expect that point P_2 is more likely. The issue in this case is that we have no information about the sampled function outside the range which is spanned by the sample set. The input corresponding to "X_1" however, is within the region for which the sample set provides information. For a new input X within the sampled region, the requirement for reduced nonlinearity tends to keep the expected output close to the outputs corresponding to known samples neighboring X. Outside the sampled region, there is no information and it is not possible to check the effect of the additional requirement against any intuitive expectancy. This of course is only an intuitive way to suggest that the requirement of reduced nonlinearity is a reasonable one. But, in any case, we should remember that beyond the consistency to the sample set, any random selection is in principle admissible, so the least argument we can make here is that introducing the additional requirement of reduced nonlinearity does not imply loss in any intuitive sense.

The requirement for reduced nonlinearity implies a reduced number of neurons in the net structure. This follows easily from the discussion regarding the net's transfer function and the form of Eq. (1), where a direct relation is established between the number of neurons and the nonlinearity inherent to the net structure. Reduced number of neurons in the net's structure is not simply an engineering optimality concern. Let us consider the function of Eq. (1) again. This function is composed in a hierarchical manner from simpler functions which are also composed from other functions in a similar way. At the lowest level, these composing functions are the neuron activation functions directly applied on the inputs of the net. The responses of the neurons at the lowest level can be viewed as representing the (degree of) presence of certain features in the input vector. These features are

combined at neurons in higher layers to synthesize more complex features on the basis of which the transfer function can be described. Thus, the number of neurons bears a direct relation to the number of features which the net synthesizes or "discovers" in order to describe a function that is consistent with the sample set. These features represent various characteristics exhibited by the training set. There is a good reason to require that the description of the target function is achieved with as few of such features as possible, or in other words, that the fitting is done with a net of as few neurons as possible. The reason is that if the number of synthesized features is restricted, then the adaptation must favor the discovery of the most powerful ones, that is those which are most useful in describing the fitting function. Thus the discovered features are more likely to reflect the "semantic" properties exhibited by the sample set. If the number of features for synthesizing the fitting function is unrestricted, then certain spurious features may be also introduced. By spurious features, we mean certain weak characteristics or properties which apply to the sample set alone but not to the sampled function. It is possible that the sample set, just because of the choice of sampling, may present some properties which are not characteristic of the sampled function. In a pattern recognition problem for instance, the samples may suggest that certain "features" are strongly correlated to the output without this being actually true for the broader object space. For example, trying to teach the concept of a "vehicle" by pointing out only blue vehicles, it may be assumed that if something is a vehicle, then it must be blue. It is difficult to decide which of the characteristics being exposed by the training set are true properties of the sampled function (let's call these *function properties/features*) and which of them are properties of the limited sample set only (let's call these *spurious properties/features*). When dealing with the problem of training a net to a function on the basis of a sample set, it is implicitly assumed in the following that the training set is typical of the sampled function and therefore it contains enough information concerning *at least* the function properties which can lead to the proper generalization. A good sample set is thought of as not being misleading, and therefore it is one which ensures that *any* possible generalization emerging from it is based on a set of properties which *at least* contains the function properties. We will also assume that within the samples of a training set, the function properties present a stronger correlation to the output than possible spurious properties do. With these hypotheses on the sample set, our intuitive goal is to identify the fitting function (generalization) which is based on (is expressible with) as few as possible of the properties revealed by the sample set. This requirement leads again to reduced number of neurons and thus reduced network nonlinearity. It should be noted that an analogy can be observed in the way humans function. If someone is presented with a set of examples of some unknown to him/her behavior

(function) which is to be inferred from those examples, then he/she will try to find the "simplest" behavior hypothesis which accounts for the observations. He/she will try to determine the smallest set of basic features which can describe adequately some behavior hypothesis.

With excessive nonlinearity, a net structure has enough "memory" so as to easily "memorize" the sample set. However, for inputs not appearing in the sample set the net may behave randomly (unpredictably) and not necessarily according to a reasonable interpolation. With a degree of nonlinearity barely enough to fit the samples, the net is forced to identify the most important collective characteristics and features which identify the sample set, thus leading to an increased likelyhood of a reasonable interpolation for inputs not appearing in the sample set. When the fitting is exact, the net is able to reconstruct the sample set. In the former case, this reconstruction is performed out of "memory" and not necessarily on the basis of "semantic" information. In the latter case, the net cannot memorize the samples one by one, so it is forced to identify a smaller set of properties or features and synthesize a compact description of the samples so as to be able to reconstruct them [9, 10]. In this case, the prediction of outputs for new inputs will be more likely based on collective characteristics exhibited by the sample set and thus a more reasonable interpolation can be expected.

4. Controlling the Development of Internal Representations

In the previous discussion, it was pointed out that the degree of nonlinearity inherent to the net structure relates to the internal representations that can possibly develop as a result of the adaptation. Furthermore, it was pointed out that one problem is to identify a net structure with a degree of nonlinearity barely enough to fit the sample set. Suppose that this problem was solved. Then given a sample set, we could identify a net for which it would be known that it can fit the samples. However, identifying the particular values of the net parameters which would achieve the exact fitting is left up to the training algorithm. Thus how exact fitting can be achieved is another problem. We discuss here the problems related to the net's internal adjustments during adaptation.

The main question here is: what could possibly prohibit the adaptive process from performing adjustments that would achieve exact fit ? This is the same problem that causes gradient descent methods to get stuck in local minima. The problem can best be illustrated by an example. Assume that we have a feed-forward network

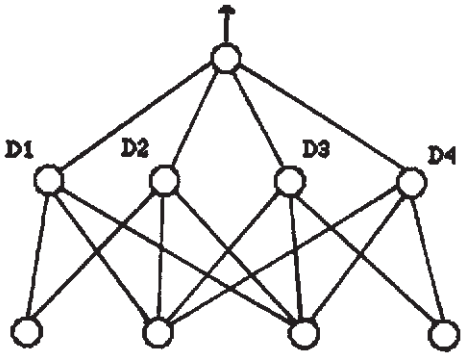

Figure 3. A specifically restricted feed-forward structure.

with a structure as in Fig. 3 having only four neurons in the lower layer and that we try to train it to a given sample set. Further assume that there exist only two sets of values for the parameters (e.g. weights) of the entire network which are such that the net's input-output function exactly fits the sample set. The net's function can be described in a closed form similar to that of Eq. (1). Each neuron's output as a function of the net's input is a descriptor for the net's transfer function. Let the descriptors represented by the neurons in the lower layer with the first set of parameter values be denoted as D_1, D_2, D_3, D_4. Respectively, let the descriptors represented with the second set of values be denoted as D_1, D_2, D_5, D_6. That is, we assume that with the given network, F_T can be represented *only* in terms of *either* the set of descriptors D_1, D_2, D_3, D_4 *or* the set D_1, D_2, D_5, D_6. Suppose now that in the course of adaptation using Back Propagation, the following case arises : one of the neurons of the lower layer adapts to D_3 and that at the same time, some other neuron adapts either to D_6 or to D_3. Indeed, it is quite possible that both neurons may adapt to D_3, because even with random initial weights two different neurons may actually end up representing the same descriptor, or in other words, they may end up responding to the same input subpattern in such a way that both responses have the same "meaning" or information content. If it so happens that during the course of the adaptation one neuron adapts to D_3 and the other adapts to either D_6 or D_3, then one of the two neurons must abandon its descriptor so that, with the given network topology, a complete set of descriptors can be identified, therefore allowing the net to properly adapt to F_T.

Usually with neural net adaptive algorithms (like the BPA), control over the weight adaptation is distributed and it is based only on local information which

comes down from higher layers. With a feed-forward structure, there is no lateral interaction among neurons in the same layer, so ties of the sort just described are quite possible and they can be responsible for cases where the net gets stuck in local minima. Of course, we used for the sake of an example an optimal net structure (no redundant neurons and thus no excessive nonlinearity) but the same remarks extend to any feed-forward structure. In feed-forward models, the lack of lateral connections impairs the quality of the developing internal representations because neurons do not effectively coordinate their adaptation. Each neuron tries to adjust its weights so as to optimize the fitting criterion on an individual basis. This does not necessarily favor the discovery of features which are mutually supportive and complementing. The result may be getting stuck in local minima, or losing "semantic" information contained in the training set. Lateral connections would allow – at least in principle – the means for neurons of the same layer to coordinate their adjustment and thus to be tuned to cooperate, that is, neurons could be tuned to produce responses which complement each other in the process of determining the proper net output. Algorithms such as the Back Propagation are based on an explicit function (the Delta rule) which determines how changes on a weight affect the change of the output. Such an explicit function can effectively be found only if there is no feedback in the net, i.e. the output signal of a neuron does not directly or indirectly affect its own input. If lateral connections are introduced then feedback is introduced and an explicit relation between the weights changes and output changes cannot practically be determined. There is an alternative though; the adaptive algorithms can be modified so as to *effect* a coordination among neurons in the same layer. This coordination would achieve the same lateral flow of information and the same goals as those possible with lateral connections. However, lateral connections need not actually (or physically) be introduced so there would be no actual signal feedback. In effect, the extended algorithms would not simply intend to optimize the output error only. The output error should be optimized *along with some other goals (terms)* which imply some sort of coordination among neurons. The key is to develop adaptive algorithms which combine features of both supervised and unsupervised learning.

Processing of an input by a feed forward net can be viewed as a series of successive transformations of the input vector to the intermediate vector output of each layer of neurons to the final net output vector. The adaptation of the net determines the particular form of the intermediate transformations. However, the intermediate successive transformations do not necessarily need to be adjusted in the same way or even concurrently. Of course each one cannot be determined independently of others. We suggest that trying to deal with all of them at the same time with

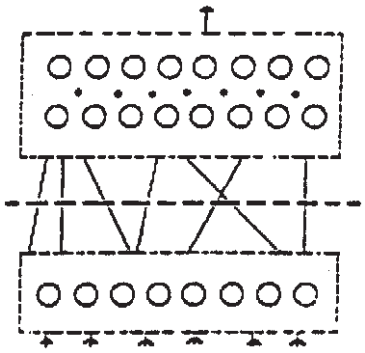

Figure 4. A feed-forward net viewed as a mapping net cascaded to a recoding net.

a single homogeneous (same for all layers) adaptive process may be an extreme way to handle the overall adaptation. As a result we loose in granularity and this can be a reason for sensitivity to initial parameters and poor quality of internal representations. Various types of adaptive techniques can be employed for adjusting the various stages of the net. Thus it is interesting to study inhomogeneous adaptive techniques by means of which attention can also be given to the development of the internal representations rather than only to fitting. Let us illustrate this idea by considering the function of a feed-forward net from this new point of view. Consider the net of Fig. 4 whose input-output mapping is some function $F : V \rightarrow R$. This net is shown separated into two nets and so it can be considered as constructed by cascading the two smaller nets. Consider now the output vector of the lower net which is input to the upper net. Let us assume that this vector spans some space R^m. Then the lower net transforms its input space V to the space R^m and the upper net performs the mapping $R^m \rightarrow R$. In other words, an internal recoding takes place. The mapping $V \rightarrow R$ is achieved by recoding the input V in terms of R^m vectors and then accomplishing the mapping $R^m \rightarrow R$. The question which arises then is, why should the two (cascaded) mappings be identified concurrently through *the same* adaptation method? The alternative we wish to consider here is the use of inhomogeneous methods combining elements of both supervised and unsupervised techniques. In particular the mapping $V \rightarrow R^m$ can be adjusted on the basis of unsupervised learning techniques whereas the mapping $R^m \rightarrow R$ can be adjusted using supervised learning techniques.

One could of course ask further, why should the upper and lower nets be of the same type (in terms of structure, squashing functions etc.)? An example of such

an alternative approach is the counter-propagation network [11] which consists of a Kohonen layer [12, 13] feeding a Grossberg layer. The Kohonen layer performs a recoding of the original input (to the net) and the Grossberg layer performs the further mapping. We are referring to this model as an example of an inhomogeneous model since it is one of the first which appeared in the literature and projected the concept of inhomogeneous net structures. However, this model is essentially a self-programming look up table and was not specifically designed to address the issue of quality internal representations. In fact, through the Kohonen layer the net's inputs are mapped to binary vectors since in the Kohonen model, neuron outputs are binary (and anyway only one neuron wins the competition). So the internally generated vectors correspond to corners of a hypercube, each of which becomes the representative internal vector for an entire region of the input space (the Voronoi region corresponding to that corner).

5. Level-By-Level Adaptation

So far our discussion and comments have been based to a large extent on intuitive concepts rather than to analytic formulations that can be of practical use. In this section, we trace the basic steps of the adaptation of a feed-forward net from an analytic point of view. This analysis provides a formal expression of the goals of the adaptation and leads to a new class of adaptive algorithms. Based on this formulation, it is possible to provide a formal definition of the net structure's optimality in terms of a formal measure.

In the following we need only consider a two layer (one hidden and one output layer) feed forward net structure. It has been shown that this structure is equally powerful with any other one of more layers [5]. We assume that the squashing function of the neurons is some fixed functional f and that the net is to be trained with a sample set which is a finite set of r pairs (X_i, Z_i), $i = 1, ..., r$, where X_i are input vectors and Z_i are their corresponding output vectors. Let us consider the structure of Fig. 5 with n input nodes, m neurons in the hidden layer and p neurons in the output layer. We denote by $\hat{X}_i$ the intermediate vector of values produced as outputs by the neurons in the hidden layer when X_i is the input. Let W_j be the weight vector associated with the jth output neuron. In the following expressions, vectors are assumed to be column vectors so the transpose of W_j, which we denote by W_j^t, is a row vector. Then the total input to the jth output neuron is the scalar product $W_j^t \hat{X}_i$. To achieve proper fitting the output $f(W_j^t \hat{X}_i)$ of

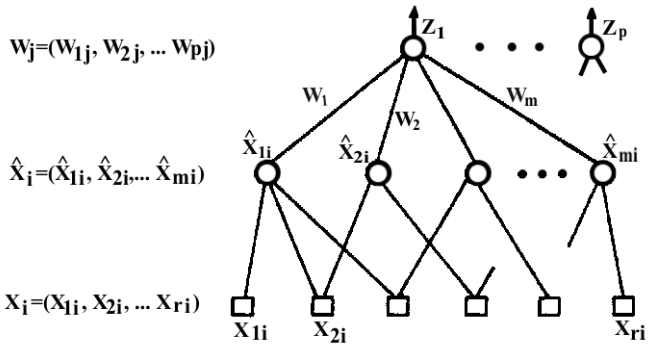

Figure 5. A n-input/p-output feed-forward net with one hidden layer.

the jth output neuron must equal Z_{ij} (the jth component of the desired output vector Z_i). Thus we obtain the following system of equations that must be satisfied :

$$f(W_j^t \hat{X}_i) = Z_{ij}, \quad \text{for} \quad i = 1..r \quad \text{and} \quad j = 1, ..., p. \tag{2}$$

We then define the values C_{ij} as $C_{ij} = f^{-1}(Z_{ij})$ and we obtain one system of equations for each output neuron as follows:

$$\text{For the j} - \text{th neuron}: \quad W_j^t \hat{X}_i = C_{ij}, \quad \text{for} \quad i = 1..r \tag{3}$$

Note that the C_{ij} values are known constants since the Z_{ij}'s and the form of the squashing function f are known. Now the above system of equations can be rewritten in the following form :

$$\text{For the } j\text{th neuron}: \quad W_j^t Y_i^j = 1, \quad \text{for} \quad i = 1, ..., r \quad \text{where} \quad Y_i^j = \frac{\hat{X}_i}{C_{ij}}. \tag{4}$$

Note that Y_i^j is a vector, and specifically, it is the intermediate vector $\hat{X}_i$ normalized by the constant C_{ij} (which can be assumed nonzero as it will be explained in the following). These normalized vectors can be partitioned in p groups of r vectors each, so that one group corresponds to each output neuron. We define the jth group G_j to be the set of vectors Y_i^j, $i = 1, ..., r$. If the net has been properly adjusted to fit the sample set (X_i, Z_i), then for each group G_j one system of equations, as per the Eq. (4), must be satisfied. Then, according to Eq. (4), all the vectors Y_i^j in the group G_j have the same projection on the direction of the corresponding vector W_j. Thus, there must exist a hyperplane normal to W_j which contains all the points whose position vectors are the vectors Y_i^j. We will use this geometric interpretation and call the point whose position vector is some vector Y as the "tip

of Y". Thus each group G_j is associated with a hyperplane (normal to W_j) which we will call *alignment hyperplane* after the fact that it contains the tips of all vectors in G_j. These hyperplanes do not need to be distinct but in general they are. So the adaptation of this net can be performed by a two step process as follows. The first step is to identify the weight vectors associated with the neurons of the hidden layer so that alignment hyperplanes exist for all groups, that is, *within each group* G_j the tips of the vectors lie in the same hyperplane – *any hyperplane*. This alignment must be *simultaneously* achieved within each and every one of the p vector groups. The second step then is to find the W_j corresponding to the alignment hyperplane of each one group G_j so as to satisfy the system of Eq. (4) corresponding to the same group.

The analysis of this section so far, focuses on the requirements or the necessary conditions of the adaptation of a feed forward net. In the following we describe a process for attaining those requirements. A lot of variations can be introduced in this process, so the following description outlines a larger class of adaptive algorithms. Such an adaptive process for a net of one output neuron only was introduced in [14] and is extended here for the general case. The task of aligning the tips of the normalized intermediate vectors can be performed as follows:

(i). (Initialization) Start with random initial values for the connection weights feeding the hidden layer. Select (at random) p hyperplanes P_j, $j = 1, ..., p$, each corresponding to a group G_j. These hyperplanes are to be transformed (orientation and position) in order to become the alignment hyperplanes at the end of the process. Each one of these hyperplanes is represented by its normal vector U_j and threshold T_j.

(ii). (Alignment) Through a repetitive process we allow the tips of the intermediate vectors to be attracted to the hyperplane (by changing the weights feeding in the hidden layer) while at the same time allowing the hyperplane to be attracted by the tips themselves (by changing U_j and T_j).

We can roughly imagine the tips of G_j as floating magnetic particles and the plane P_j as a free moving metal sheet. The sheet attracts the particles which move toward it (changing the weights of the hidden layer). At the same time the sheet is attracted by the particles and moves toward them by changing its position and orientation (changing U_j and T_j). This effect can be achieved by minimizing the sum of the square distances of a set of points from a hyperplane. Thus we define the distance measure corresponding to a group G_j as: $D_j = \sum_i^r (U_j^t Y_i^j - T)^2$. Thus the goodness of the overall alignment can be measured by $D = \sum_j D_j$. Since all the

D_j's are positive definite, minimization of D requires the simultaneous minimization of all the D_j's. Overall alignment is complete when $D = 0$. Minimization of D is to be achieved by changing the weights of the hidden layer as well as changing the U_j's and T_j's according to either a gradient descent procedure ($\Delta W = -\alpha \frac{\partial D}{\partial W}$) or a simulated annealing method [4]. As it will be explained later, a solution exists if there are enough neurons in the hidden layer to allow sufficient flexibility in the transformation (movement) of the normalized intermediate vectors. However, the minimum number of neurons required for overall alignment is not known a-priori. So one way around this problem is to form the net incrementally. Thus the process of step (ii) above can be augmented as follows :

(iii). (Expansion) Start with one hidden layer and a fixed number of neurons. Using step (ii) minimize D. If D cannot be reduced to 0 then add one more neuron in the hidden layer and continue until $D = 0$.

According to step (iii), the net structure is started off with a relatively small number of neurons and is gradually expanded. Before each step of the expansion, the best adjustment of the hidden layer weights is found in order to bring the vector tips of each group G_j as close as possible to being on a corresponding hyperplane. If there are not enough neurons in the hidden layer, then the implied transformation of the original input vectors to the intermediate normalized vectors will not have enough nonlinearity to move the normalized vectors to the desired alignment. So the hidden layer is expanded and the process continues. There is no need to start the adaptation all over after each step of expansion. Two practical considerations can be added here. The first one concerns the time of the decision to expand the hidden layer. Expansion is deemed necessary if D cannot be reduced further but it may be more practical to do it if the rate of reduction becomes unacceptably slow. The second consideration concerns the size of the expansion. It would be best to add one neuron at each step of the expansion but it may be more practical to add more than one neurons in the early steps.

After the alignment is complete, the weight vectors for the output neurons must be determined. Thus the final step is:

(iv). (Find W_j's) After alignment the weight vector W_j of the jth output neuron is determined by solving the system (4) corresponding to G_j.

There is a fine point for step (iv). Before explaining this point we need a couple of definitions. We define the rank of a system (4) to be the number of its linearly independent equations. Let us denote by S_j the set of vectors which is obtained if

each vector Y_i^j of the group G_j is augmented by one more component which is set to the constant 1. We then define the rank(S_j) to be the number of independent vectors in S_j, which is obviously the same as the rank of the corresponding system (4). Now let us see why step (iv) above is necessary. The way the alignment was performed required the use of vectors U_j for representing the alignment hyperplanes P_j. So it would appear that the W_j should be colinear to U_j and thus the solutions are the U_j's scaled appropriately so as to satisfy the corresponding systems (4). This scaling is a trivial task and does not require solving each system entirely. Suppose however, that after the alignment is complete the hidden layer ends up with m neurons and that the rank of the system corresponding to some group G_j is ℓ where $\ell < m$. Then there are infinite solutions for that system. In other words, the vectors of G_j are in R^m but only ℓ of them are linearly independent, so there are infinitely many planes on which the vectors are already aligned. Thus, there exists a solution for W_j with $\ell - m$ zero components. A zero value for the i-th component of W_j means that a connection from the i-th hidden neuron to the jth output neuron is not necessary. Thus each system must be solved, by using Gaussian elimination which is quite efficient, in order to identify the W_j vectors so that each one has the greatest possible number of zero components.

At this point, the outline of the adaptation is concluded. A couple of remarks for enhancing step (i) can be added though. It was mentioned that the initial hypeplanes P_j are selected at random which means random positions/orientations. A better choice however, is to start them off close to the initial vector tips that each is supposed to align. This can easily be achieved with a simple regression.

The most important differences of this approach and the Back Propagation is that the adaptation proceeds layer by layer in a forward manner. This allows the possibility to change the number of neurons in the hidden layer during the adaptation. This cannot be done quite effectively in Back Propagation which is very sensitive to changes of the net structure during adaptation. Also, with the proposed approach there is far better control on the adjustment of the weights because each weight directly affects the optimality measure (minimization of D). In contrast, with the Back Propagation weights of intermediate layers affect transitively the error measure which is to be optimized because the signal flowing through a connection (synapse) is altered by other connections (whose weights are to be updated also) on the path to the output. In our suggested approach, there are combined characteristics of both supervised and unsupervised techniques. The adaptation does not target only the fitting. Fitting is attained by manipulating directly the internal representations. This type of operation is in line with the discussion of section 4. Aligning the intermediate vectors can be viewed as unsupervised learning

then followed by supervised learning at the output layer.

6. Discussion

The formulation which was developed in the previous section gives rise to some interesting remarks. Since there are m neurons in the hidden layer, the vectors $\hat{X}_i$, Y_i^j, and W_j, are in the space R^m. If $m \geq r$, each group G_j represents exactly r points (vector tips) in R^m and there always exist alignment hyperplanes for all groups. In other words with $m \geq r$ all the systems (4) corresponding to the various groups can be simultaneously solvable. There exists at least one alignment hyperplane for each group even if it is assumed that the rank of each system (4) is (the maximum possible) r. Of course, this is a trivial solution and requires at least one neuron in the hidden layer for each input vector of the training set. It virtually corresponds to a hidden layer capable of "memorizing" each of the training set vectors individually and the net then functions as a look-up table, thus poor generalization capability is expected. We are interested in the minimal required number of hidden layer neurons. Although this is an open question, we can at least define a lower bound in terms of a compact expression. From the above formulation in terms of systems of equations, as in Eq. (4), it follows immediately that m should be at least as large as the rank of any one of the systems (4). Thus, after all groups G_j have been aligned (and all systems (4) are satisfiable), the minimal number of hidden neurons should be equal to the greatest one of the ranks of these systems. The ranks of those systems depend on the choice of the weights associated with the connections between the input layer and the hidden layer. So, as far as the adjustment of these latter weights is concerned, the goal of the adaptation is to minimize the ranks of the systems (4). Thus the adjustment of the connection weights feeding the hidden layer should aim to minimize the meassure: $max_j\ [rank(S_j)]$. Let $min_W\ [rank(S_j)]$ be the minimum that the $rank(S_j)$ can assume over the space of the weights feeding the hidden layer. Threfore, a lower bound for m is $max_j\ [min_W\ [rank(S_j)]]$, indicating the minimum number of neurons required for proper alignment of *any* of the groups. However, this lower bound is a direct consequence of the fact that a single network is used for producing all the components of the desired output vector Z_i. The same intermediate vectors $\hat{X}_i$ must satisfy the systems (3) for all groups G_j simultaneously. If a different network was used for producing each one of the components of the output vector, then the intermediate vectors developed in each net for the same input need not be the same and we would only need to minimize the rank of a single system (4). Since the rank associated with each net should be

minimized independently of others, it is possible that (after the said minimization) the maximum one of the individual ranks obtained can be smaller than the lower bound obtained above for a single net. It thus turns out that it is preferable to use a collection of nets rather than a single net because the goals of the adaptation, as analyzed in sections 3 and 4, can be served better. In fact, there is probably only one reason for which we might prefer a single net of multiple outputs. Such a net implements many transfer functions (one for each output neuron) on the same structure. All these functions use common descriptors (or at least they are very likely to). Thus a single net may be preferable when we wish the various functions corresponding to the various outputs to be based on as many common descriptors as possible.

It can also be seen that the form of the squashing function of the output neurons (but only those specific ones) is not really critical in the above process. Of course, the particular C_{ij} values affect the solutions but the approach does not depend on how they were obtained. So for that matter, the squashing function of the output neuron could as well be linear and thus the output neuron can just perform addition. That would have the additional advantage of not imposing bounds on the range of the output values like the sigmoid function does. In fact, the squashing function for the output neurons can be chosen so as to ensure that no C_{ij} value is zero in order to alleviate problems in the normalization of Eq. (4). A convenient choice for example, would be a linear function with a constant bias $f = W^t X + b$, where the bias b is some constant which ensures that $W^t X$ is nonzero in any instance of the sample set (and only for the instances of the sample set). For the jth output neuron for example, we can use $f_j = W_j^t X + b_j$, where $b_j \neq Z_{ij}$ for $i = 1, ..., r$. If the output neuron is linear, then the net implements a transfer function similar to the functional which Kolmogorov had shown capable of approximating every continuous real function of many variables [15]. Kolmogorov had proven this important result but the proof was not constructive (i.e. it did not provide a method for identifying the components of the functional) and thus his theorem could not be of use for practical purposes.

7. Conclusion

Controlling the nonlinearity of the net structure as well as controlling the development of internal representations during adaptation are issues implicitly linked to a feed forward model's robustness and capability to capture semantic information exhibited in the training set. The discussion of this paper linked these issues

to a new formulation of the adaptation process. We feel that the most significant contributions of this paper are the new formal analysis of the adaptation and the new adaptive method. The analysis allows the formal characterization of the goals of the adaptation in terms of well defined measures and provides useful insights concerning the problems of the adaptation. These insights are significant in any effort to develop adaptive algorithms for feed-forward nets. The significance of the adaptive method is that it allows control on the internal representations, and the flexibility to accommodate incrementally formed nets.

References

[1] R.B. Banerji, *Artificial Intelligence: A Theoretical Approach*, Elsevier/North Holland Scientific Publishers, 1980

[2] D.E. Rumelhart, G.E. Hinton and R.J. Williams *Learning Internal Representations by Error Propagation*, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol1: Foundations*, D.E. Rumelhart and J.L. McClelland (editors), MIT Press, Cambridge, MA (1986)

[3] S. Kirkpatrick, C.D. Gellat Jr. and M.P. Vecchi, *Optimization by Simulated Annealing*, Science, Vol. 220, No. 4598, May 13, 1983.

[4] H. Szu and R. Hartley, *Fast Simulated Annealing*, Physics Letters A, Vol. 122, No. 3, 4, June 8, 1987.

[5] K.I. Funahashi, *On the Approximate Realization of Continuous Mappings by Neural Networks*, Neural Networks, Vol.2, 1989

[6] L.J. Han, Van der Maas, Paul F.M.J. Verschure, and Peter C.M. Molenaar, *A Note on Chaotic Behavior in Simple Neural Networks*, Neural Networks Journal, Vol. 3, pp. 119-122, 1990.

[7] J.F. Kolen, and J.B. Pollack, *Backpropagation is Sensitive to Initial Conditions*, Complex Systems, V.4, 1990

[8] M.L. Brady, R. Raghavan, and J. Slawny, *Back Propagation Fails to Separate Where Perceptrons Succeed*, *IEEE Trans. on Circuits and Systems* v.36, no5, May 1989.

[9] C. Koutsougeras and C.A. Papachristou, *Training of a Neural Network Model for Pattern Classification Based on an Entropy Measure*, in Proc. IEEE Internat. Conf. on Neural Networks (ICNN '88), New York: IEEE, July 1988.

[10] C. Koutsougeras and C.A. Papachristou, *A Neural Network Model for Discrete Mappings*, in Proc. IEEE Internat. Conf. on Languages for Automation (LFA '88), New York: IEEE, October 1988.

[11] R. Hecht-Nielsen, *Neurocomputing*, Addison-Wesley Publishing Co., 1990

[12] T. Kohonen, *Representation of sensory information in self-organizing feature maps, and relation of these maps to distributed memory networks*, SPIE Vol. 634 Optical and Hybrid Computing(1986), pp. 248-259.

[13] T. Kohonen, *Self-organized formation of topologically correct feature maps*, Biological Cybernetics 43:59-69

[14] C. Koutsougeras and G. Papadourakis, *A Method for Training a Feed-Forward Neural Net Model While Targeting Reduced Nonlinearity*, Proceedings ICTAI '91 international conference, San Jose, CA, Nov. 1991.

[15] A.N. Kolmogorov, *On The Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition*, American Mathematical Society Transactions, 1963, 28, 55-59

[16] C.L. Scofield, *Learning Internal Representations in the Coulomb Energy Network*, in Proc. IEEE Internat. Conf. on Neural Networks (ICNN '88), New York: IEEE, July 1988.