A Method for Training a Feed-Forward Neural Net Model While Targeting Reduced Nonlinearity

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

George Papadourakis
Department of Computer Science
University of Crete
P.O. Box 1470, Heraklion, Crete
Greece

Abstract

An analysis presented here for feed forward neural nets examines the causes of problems in the adaptation of current models. A new method for training a feed forward neural net model is introduced here. The method encompasses elements of both supervised and unsupervised learning. The development of internal representations is no more an issue tangential to the curve fitting objectives of the other known supervised learning methods. Curve fitting remains as a primary objective but unsupervised learning techniques are also employed in order to aid the development of internal representations. The net structure is incrementally formed thus allowing the formation of a structure of reduced nonlinearity.

1. Introduction

This research is motivated by the need for neural net models with dependable generalizing capabilities. With current models the goals of the adaptation are limited and focused on fitting the training set per se. The development of the internal representations is up to random parameters or factors of the training process as long as a variety of different representations can result in fitting the training set.

In general, the adaptation of a supervised feed-forward net is a curve fitting process. The training set used for the adaptation is obtained by sampling a target function which the net is to "learn". Current methods base the adaptation on the optimization of some measure $E_r$ for the fitness as for example the output square error. However, to base the adaptation on such a measure of fitness alone is not an adequate approach because it leaves some important factors up to random decision. For example, the network topology (number of layers and neurons per layer), initial weights, learning rate, momentum parameters, etc. are chosen almost at random, yet these parameters significantly affect the end result of the adaptation. Current models are very sensitive to the choice of training parameters and although being able to learn the training set per se, they may not behave properly outside that set. This impairs their robustness and reliability and thus the utility of this new technology.

Our approach focuses both on fitting the training set and on the quality of the internal representation which relates to the models ability to derive generalizations that capture the "semantic" information contained in the training set. Thus this research is based on the view that learning a function is not simply a matter of curve fitting on the training set without attention to the internal representations which develop as a result. Increase of robustness is the result of the adaptation's independence from the choice of initial parameters (a very serious problem in most current models) and also due to increased likelihood that the adaptation is based on semantically important features of the input.

Learning is approached here by means of an adaptive algorithm which combines elements of supervised and unsupervised methods. 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. We present here the view that the function of a net is basically a two stage process and that there are advantages to adaptive techniques which treat it as such. We suggest that the two stages of the adaptation 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.
In the following we analyze the functioning of a general feed forward net and point out the problems which current training methods entail. We then discuss 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 present a new training method which combines properties of both types of learning. The analysis on which this training method is founded provides a base for the further study of optimal nonlinear nets.

2. Background

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 to that neuron's input. For such networks without feedback, the topology can be thought as representing a dependency graph. Neurons are arranged in layers so that the total input stimulus to each neuron depends only on neurons of 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 function of the net shown in figure 1. Let \( F \) be the net’s output, \( O_i \) and \( f_i \) be respectively the output and the activation or squashing function of neuron \( n_i \), and let \( W_{ij} \) be the connection strength or weight from \( n_i \) to \( n_j \). The total net output being the output of neuron \( n_k \) can be expressed as:

\[
F = O_k = f_k(\text{net input}_k) = f_k(W_{64}O_4, W_{56}O_5) = f_6(W_{46}f_4(W_{14}O_1, W_{24}O_2, W_{34}O_3), W_{56}f_5(W_{25}O_2, W_{35}O_3))
\] (1)

The above equation shows that with such a feed-forward structure the net’s function is a composition of the simple activation functions of the net’s neurons. The number of layers defines the depth to which the activation functions are embedded. The above function can be generalized to feed-forward nets with arbitrary numbers of layers and neurons per layer. In the following we will refer to the specific function instances which synthesize \( F \) in equation (1) as descriptors or higher order features. For example in equation (1) \( f_4(W_{14}O_1, W_{24}O_2, W_{34}O_3) \) and \( f_5(W_{25}O_2, W_{35}O_3) \) are descriptors for \( F \). They can also be viewed as higher order features composed of the input features or from other lower order features. 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 inputs (as opposed to the neuron’s inputs).

![Figure 1](image-url)
simulates a gradient descent according to the rule:
\[ \Delta W = -\alpha \frac{\partial E}{\partial W} \]
known as the Delta rule.

The adaptive process terminates when \( E_q \) cannot be reduced further. There is of course the question of this gradient descent getting stuck in local minima of \( E_q \). What is a hard problem though is determining the net’s topology which is chosen apriori. One must virtually guess the necessary number of layers and neurons per layer which would facilitate proper approximation of a specific target function \( M \) 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 \( M \). 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 \( f_1 \)’s). However, the number of neurons per layer is a serious question.

The choice of topology (structure) for a given \( M \) 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 \( E_q \) 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 \( E_q \) can be attained. This means that the net’s \( F \) coincides with the target \( M \) at least for the points of the training set, but there is no guarantee that \( M \) is approximated properly in other points. In other words, a 0 for \( E_q \) may imply the possibility that the net contains more nonlinearity than is exactly necessary.

The 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. A curve fitting approach which is based only on the objective of discovering a surface which contains the sample points, will at best select randomly one of the \( S_i \)’s.

If there is excessive nonlinearity then it is possible to represent many different functions all of which fit exactly the training set. Which is the one to be identified by the adaptive mechanism is random and it is heavily dependent on various parameters of the adaptation mechanism such as initial weight values, step size (reflected by the learning rate \( \alpha \)), momentum etc. Thus there is a strong potential to obtain various functions \( F \) with the same training set depending on the particular choice of initial random parameters such as the initial weights. Reports [6][11] 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. 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?).

The inability to determine a net topology appropriate for a given \( M \) is the major difficulty with the feed-forward model. It would seem then that leaving the development of the net’s structure up to the adaptive process is a promising solution. Then the net’s nonlinearity could be controlled since in the course of the adaptation neurons can be added or deleted. In this way a net topology or structure can be identified bearing just enough nonlinearity to properly fit the training set. This of course means an optimal total number of neurons on the net as well. Why do we target reduced number of neurons? It is of course because they reflect the degree of nonlinearity in the net, but there are other reasons as well. Referring to the net’s nonlinearity is a way to explain the problems of the adaptation in an abstract way but other points of view converge into the same requirements of reduced number of neurons. The fewer the neurons in the net the fewer the descriptors used to synthesize the learned input-output function and the better the chance to avoid "learning" a function which really reflects overgeneralization.

An analogy can be observed with the way humans function. If a human is presented with a set of examples of some unknown to him/her behavior (function) which is to be inferred from those examples, the human 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 on the basis of which some behavior hypothesis may be developed. Current neural net models will not necessarily do that; in many cases the set of features they discover is random as long as the example set is accounted for. Consider for example a second order polynomial which is sampled with sufficient density by a finite training set of \( n \) samples. If \( n \) is sufficiently large the shape of the sampled polynomial could be quite apparent (and a human subject would then most likely assume that the sampled function is a second degree polynomial - the one of least nonlinearity which can properly fit the samples). However, the exact same samples could have been obtained by a particular way of
sampling a specific polynomial of degree \( n \) or higher which fits the given samples. Whether the intended function or some other was determined by the adaptive mechanism of the net depends on how well the mechanism captures "semantic" properties exhibited by the training set as a whole. In many cases the training set samples, just because of the choice of sampling, may present some properties which are not characteristic of the target function. Using the reverse argument of the same example as above, a 10th degree polynomial may be sampled in such a way that the sample set may suggest that the sampled function is a 2nd degree polynomial. Or, in a pattern recognition problem for instance, the training samples may suggest that certain "features" are strongly correlated to the output without this being true. 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 exposed by the training set are true properties of the sampled function (let's call these \textit{function properties/features}) and which of them are properties of the limited sample set only (let's call these \textit{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 that therefore it contains enough information concerning \textit{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 \textit{easy} possible generalization emerging from it is based on a set of properties which \textit{at least} contains the function properties. We will also assume that with 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 generalization (function) which is based on (is expressible with) as few as possible of the properties revealed by the sample set and this requirement also leads to reduced number of neurons.

To achieve a net structure of optimal nonlinearity a method which seem to be a good solution is to incrementally form the net by adding neurons as needed. However, it is not at all simple to directly allow incremental formation of the net structure in a feed forward model which uses Back Propagation for adaptation. The Delta rule explicitly connects a weight parameter to layers which follow the one directly affected by that weight [9]. Thus the entire net structure must be known in advance and further changes in the topology introduce problems.

Other reports [2] also refer to the inability of the Back Propagation to guarantee optimal weights to minimize the output error. In [2] it is shown that Back Propagation may fail to perform properly for some linearly separable cases in which the old perceptron algorithm would succeed. In order to understand the difficulty which is inherently associated with the adaptation of a general feed-forward network, let us consider the following simplistic example. Assume that we have a classic feed-forward network as in figure 2 with a given topology having only four neurons in the lower layer and that we try to train it to a certain input-output mapping \( M \). 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 network's input-output function exactly matches \( M \). 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, \( M \) can be represented \textit{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 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 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 \( M \).

![Figure 2](image-url)

With an algorithm like the BPA, control over the weight adaptation is distributed and it is based only on local information which comes down from higher layers. 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. This also is the reason for the success of competitive learning nets and the attention they have attracted lately. In such models there is interaction among neurons in the same layer (such as in the "winner take all" approach) which automatically breaks such ties.

3. Controlling the Internal Representations

The analysis so far motivates the goals of developing adaptive algorithms which exercise some sort of control over the formation of internal representations during training. In this way we seek to increase the probability that semantically important descriptors of the target function (function properties) will be identified during the adaptation. Also, when possible we seek to keep the degree of nonlinearity inherent to the net reduced (mostly by developing the net incrementally). Our method concerning adaptation of feed forward nets revolve around two main ideas which can be sketched as follows:

1. In feed forward models the lack of lateral connections impairs the quality of their 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. That does not necessarily favor the discovery of features which are mutually supportive and complementing. The result may be getting stuck in local minima, or loosing "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 affects 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 weight and output changes cannot practically be determined. What we propose here is to modify the adaptive algorithms so as to effect a coordination among neurons in the same layer. This coordination will achieve the same goals as those possible with lateral connections. However, lateral connections will not actually (or physically) be introduced so there would be no actual signal feedback. In effect the extended algorithms will not simply intend to optimize the output error only. The output error will be optimized along with some other goals which imply some sort of coordination among neurons. The key is to develop adaptive algorithms which combine features of both supervised and unsupervised learning.

2. 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, each one of the intermediate successive transformations do not necessarily need to be adjusted at the same time. 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 a single homogeneous (same for all layers) adaptive process may be an extreme way to handle the overall adaptation. We lose in granularity and this can be a reason for sensitivity to initial parameters and poor quality of internal representations. We suggest here that 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 the fitting.

4. Level-by-level adaptation

Let us now view the function of a feed-forward net from a different point of view. Consider the net of figure 3 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's 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 re-coding takes place. The mapping \( V \rightarrow R \) is achieved by re-coding 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? Further, why should the upper and lower nets be of the same type (in terms of structure, squashing functions etc.)? The alternative we wish to consider here is the use of inhomogeneous methods containing elements of both supervised and unsupervised learning.
unsupervised techniques. In particular the mapping $V \to R^n$ can be adjusted on the basis of unsupervised learning techniques whereas the mapping $R^n \to R$ is adjusted using supervised learning techniques.

\[ W = (W_1, W_2, \ldots, W_m) \]
\[ X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \]
\[ X_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \]

**Figure 3**

Let us assume a two layer (one hidden and one output layer) feedback net with a single neuron in the output layer as shown in figure 4. We also assume that the squashing function of the neurons is some fixed functional $f$ and that the net is to be trained with a training set which is a finite set of pairs $(X_i, Z_i)$ where $X_i, i=1..r$ are input vectors and $Z_i$ are their corresponding output values. Let us 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. Then the net input to the output neuron is $\hat{X}_i W$, where $W$ is the weight vector associated with the inputs of the output neuron. Since $f(\hat{X}_i W)$ must equal $Z_i$ we obtain the system of equations:

\[ \hat{X}_i W = C_i, \text{ for } i=1..r \text{ where } C_i = f^{-1}(Z_i). \]

Note that the $C_i$ values are known constants since the $Z_i$'s and the form of the squashing function $f$ is known. Now the above system of equations can be rewritten in the following form:

\[ Y_i W = 1, \text{ for } i=1..r \text{ where } Y_i = \frac{\hat{X}_i}{C_i} \]

According to the above equations all the vectors $Y_i$ have the same projection on the direction of the vector $W$. Thus there must exist a hyperplane which contains all the points whose position vectors are the vectors $Y_i$. Let us use this geometric interpretation and call the point whose position vector is a vector $Y$ as the 'tip of $Y$'.

Thus if the net has been properly adjusted during adaption, then the tips of the vectors $Y_i$ (which are the $\hat{X}_i$'s normalized by their corresponding $C_i$ values) must lie on some hyperplane and the vector $W$ is then normal to that hyperplane. So the adaptation of this net can be performed by a two step process as follows. First, identify the weight vectors associated with the neurons of the hidden layer so that the tips of the vectors $Y_i$ lie in the same hyperplane - any hyperplane. Then find the $W$ which is normal to that plane and its measure achieves a scalar product of 1 according to equation (3).

Before we proceed with the outline of the methods which implement the first step, let us present some remarks derived from the above formulation. Assume that in the hidden layer there are $m$ neurons. Thus the vectors $\hat{X}_i$ and $W$ are in the space $R^m$. If $m \geq r$ then there always exist at least one hyperplane for the solution no matter what the choice for the weight vectors of the hidden layer (assuming that the rank of the system (3) is $r$). 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 but poor generalization capability should be expected then. The question which we have to address in this research is: what is the minimal number of hidden layer neurons required for the existence of a solution hyperplane and what is an efficient way of determining that number? This is equivalent to asking what is the rank of the system (2) and then take it from there.

It can also be seen that the form of the squashing function of the output neuron (but only that specific one) is unimportant since its role in the above process is not really critical. Of course the particular $C_i$ values affect the solutions but the approach does not depend on how they were obtained. So far for that matter the squashing function of the output neuron could as well be linear and thus the output neuron can just perform addition.
6. Conclusion

Controlling the nonlinearity of the net structure is an issue implicitly linked to a feed forward model's robustness and capability to capture semantic information exhibited in the training set. We discussed here a model which allows control on its nonlinearity. Although optimality is not always guaranteed, the approach founded with this model provides a basis to pursue optimal nets.

References

would have the additional advantage of not imposing bounds on the range of the output values like the sigmoid function does. 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. 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 engineering purposes.

We now return to the identification of the hidden layer's weight vectors which bring the tips of the normalized intermediate vectors on the same hyperplane. A number of analytic approaches can be employed which however would be computationally intense. However, a simple iterative approach can be employed as follows:

i. Start with some random initial weights and some random hyperplane represented by its normal vector \( w \) and threshold \( T \). Then in a repetitive process allow the tips of the intermediate vectors to be attracted to the hyperplane (by changing the weights feeding in the intermediate layer) while at the same time allowing the hyperplane to be attracted by the tips themselves (by changing \( w \) and \( T \)).

Roughly imagine that the tips are floating little magnets and the plane is a free moving metal sheet. This effect could be achieved by minimizing the sum of the square distances from the hyperplane: \( D = \sum_i (T_i w - T)^2 \). The minimization of \( D \) is to be achieved by changing the weights of the intermediate layer as well as changing \( w \) according to either a gradient descent procedure or a simulated annealing. Solution will exist if the number of neurons is at least as large as the rank of the system \( 2 \).

What is the minimum rank for the system \( 2 \) which can be achieved is yet an open problem and we will address it in future work. The answer to this question will yield optimal nets. Currently we address reduced rank for \( 2 \). To this end we elude to forming a net incrementally which will achieve the goal of flattening the tips of the intermediate normalized vectors (bringing them on a single hyperplane. This is done by adding layers one at a time as needed. Thus some more steps are added to the original algorithm as follows:

ii. One can start with one layer and a fixed number of neurons and try to do the best adjustment on the weights associated with it in order to bring the tips of the normalized intermediate vectors as close to being on a plane as possible. The previous method (i) is used to this end. Then those weights can be fixed and another layer added to the net. Then step (i) is repeated for the weights associated with the new layer in order to adjust the new weights.

iii. The addition of layers can continue until the goal of flattening the tips of the normalized vectors is achieved. The normalization at each step can be done with the same constants since any hyperplane will provide a solution as long as it encounters all the tips. Then the weight vector of the output neuron is determined through system \( 2 \).

Variations to this approach can also be employed. The method can be enhanced with mechanisms for identifying the initial parameters such as the weights of a newly added layer and the starting position/orientation of the hyperplane. This is easily achieved with a simple regression.

The most important differences of this approach and the Back Propagation is that 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 our approach there is far better control on the adjustment of the weights because each weight directly affects the function to be optimized. In contrast, with the Back Propagation weights of intermediate layers affect transitively the output function which is to be optimized because the signal flowing through a connection (synapse) is altered by many other connections (whose weights are to be updated also) in the path to the output. In our suggested approach there are combined characteristics of both supervised and unsupervised techniques. Bringing the tips to a plane can be viewed as unsupervised learning then followed by supervised learning of the output layer.

5. Discussion

Why is it a good idea to use unsupervised learning then followed by supervised learning? Because in this way the quality of the internal representations will be much better. In current models the development of the internal representations during the training phase is driven by the demand to strictly optimize the error but what guarantees are there that this demand alone will always provide the drive for the discovery of good features? Of course the demand to optimize the error will make it obvious whether some feature that happens to be there is good or not but in principle there is no reason to believe that this demand will always foster the synthesis (through the