AN INTRODUCTION TO NEURAL NETWORKS BASED ON THE FEED FORWARD, BACK PROPAGATION ERROR CORRECTION NETWORK WITH WEIGHT SPACE LIMITING BASED ON A PRIORI KNOWLEDGE

William E. Blass, Department of Physics and Astronomy and The University of Tennessee Computing Center and Paul B. Crilly Department of Electrical and Computer Engineering The University of Tennessee Knoxville, TN 37996

Abstract

Neural networks will be introduced for instrumentation professionals at an introductory level. The structure of neural networks will be described with particular attention paid to the back propagation network. Both graphic and analytical descriptions will be used to efficiently bring the listeners into the picture. Examples of back propagation networks applied to one and two dimensional resolution enhancement will be used to exhibit characteristics of the networks. In the two dimensional case, image recovery and enhancement of Hubble Space Telescope-like images will be used as examples.

While nearly all deployed neural network solutions to real problems use one form or another of the back propagation network, training the network can be very time consuming. We have searched for and found several approaches to the effective limitation of the network weight space (sometimes thought of as the networks long-term memory). The conceptual basis of weight space limitation will be introduced. The connection of weight space limitation to incorporation of a priori knowledge of the systems to which the networks are applied will be discussed with examples.

The goal of the tutorial will be the introduction of those with no or only a passing acquaintance with neural networks to the world of back propagation networks applied to non-trivial real world problems. The examples will be drawn from exploratory work on generalization in neural networks.

Introduction

Neural Networks, as brain models and as computational models have become very popular in the last several years. The history of neural networks is full and varied. The emergence of several professional societies and several more archival journals than societies in the last few years makes it difficult to do justice in a short overview of the field. Suffice it to say that if the reader is unaware of neural network brain and/or computational models we will be very surprised. The field is unusual for a "research" field in that commercial applications have been marketed and "installed" already. A DARPA administrator was once said to have compared the importance of the emergence of neural networks to the importance of the development of nuclear fission reactors. Nonetheless, peeling back the hype and looking inside, one finds much of interest about neural networks [1-5]. An overview of sorts is provided by the DARPA study [6].

Our interest in neural networks is enmeshed in our interest in artificial intelligence and also in new and perhaps useful computational paradigms. Our expertise is not in the area of neuro-physiology and brain models and so tend to leave that aspect of neural networks to other more qualified investigators. It occurred to one of us that feed forward, back-propagation networks, since they were fairly successful at pattern recognition and somewhat more noise immune that one would have predicted, might be able to engage in a primitive form of generalization. One of us had done quite a bit of work in the area of resolution enhancement (deconvolution) of high resolution infrared spectra [7-9]. Since to a limited extent one could see "pattern recognition" aspects of deconvolution, it was decided to try to teach a network to deconvolve spectra. The reasoning was that if neural networks could generalize at all, then they should be able to be taught to deconvolve spectra at some level of competence. It should be emphasized that we have no interest or intent to use neural networks to do production deconvolution (unless of course they prove superior to the systems in place). The project is a test-bed for generalization in neural networks and an introductory, real world application to whet appetites for more understanding.

What is a neural network? The best way to understand a neural network is from an operational point of view. In order to do this we employ the notion of a neural network as a black box -- in much the same sense that for centuries (and many would argue -- yet today) the human brain was best viewed operationally as a black box.

Fig. 1. The neural network as black box. In the parlance of system theory, network in is the input signal to the system seen as black box -- here a neural network. The output from the neural network as black box is the network out signal.

In order to appreciate what a neural network does, and thereby grasp its definition in an operational sense, consider the following example.

Many physical measurement systems are modeled as linear systems. In this sense these systems may also be viewed as an operationally defined black box as in Fig. 2.

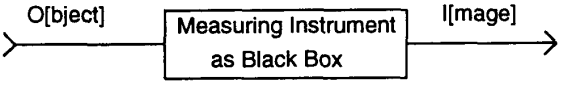

Fig. 2. The measuring instrument as black box. In the parlance of the image processors and optical imaging practitioners, the instrument input is referred to as the object or source of input (photons from the object) while the instrument output is referred to as image or the transformed output (transformed photons processed by the optical system).

Modeled as a linear system, the instrument as black box is characterized by an instrumental response function R. The linear systems model yields

$$I = O \otimes R \tag{1}$$

where $\otimes$ represents the convolution operation.

As an example, consider high resolution spectroscopy of gas phase molecules. In the simplest case, one uses a dispersive spectrometer and observes the depletion of the incident photon beam due to absorption by gas sample. Assuming the gas sample is a low pressure sample, and that the maximum absorptance is in the 0.2 to 0.3 range (20 to 30% absorption), then the true spectrum is well modeled by a collection of co-added Gaussian spectral lines whose areas are a linear function of the intensities of the transitions. An example of such a spectrum is shown in the third trace from the top in Fig. 3. After passing through a spectrometer, the true and desired data characterized by trace 3 in Fig. 3 is observed as trace 2 in Fig. 3. The broadening of the transition profiles as observed is due to the restrictive effects of the instrumental response function R in Eq. (1).

One dimensional resolution enhancement or deconvolution has as its goal the recovery of the lost resolution and hence the lost information in the observed as opposed to the true spectrum. In terms of the linear

0-7803-0640-6/92 $3.00 © 1992 IEEE

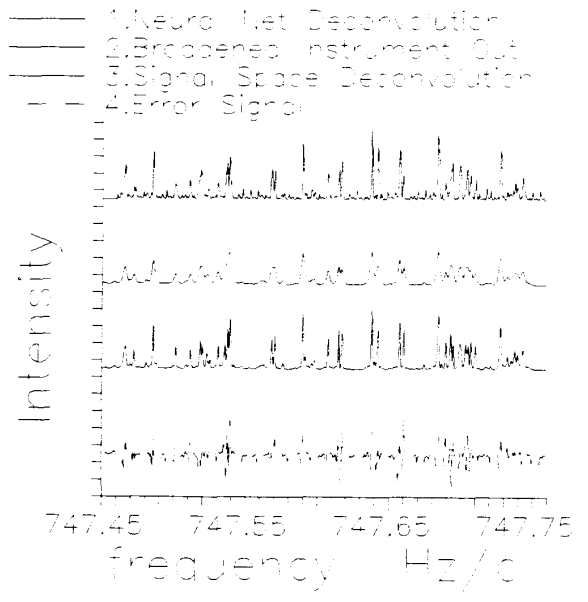

Fig. 3. Sample of the deconvolution of a laser spectrum of propane. The legend refers to the traces from top to bottom. The fourth trace (4) is the difference between the neural network output in trace 3 when presented with the observed data in trace 2. Trace 3 is actually the result of a signal space deconvolution of the experimental data (trace 2). In the text trace 3 is presumed to be the equivalent of the true data observed by the laser spectrometer for purposes of the discussion.

systems model of Fig. 2, the true spectrum (trace 3, Fig. 3) is the Object, and the output of the spectrometer (trace 2, Fig. 3) is the Image of Fig. 2. This sort of recovery is readily carried out (with good signal to noise data) using a signal space technique proposed by Jansson and described in detail by Blass and Halsey [7]. This enhancement technique has been extensively tested and is a production technique where signal to noise in the data is on the order of 100:1 and where the instrumental response function can be measured or readily modeled.

The spectrum of Fig. 3 is characterized by a Gaussian of width (in data points) 6 for trace 2 and of width 2 for trace 3. We would like to provide trace 2, the observed broadened data, to the neural network of Fig. 1 and have the output of the network give us the equivalent of trace 3 -- the true and desired spectral data. The iterative signal space technique of Jansson does this quite reliably without creating artifacts and without losing information.

The data in Fig. 3 is a tunable diode laser spectrum of propane gas as is thought to exist in the atmosphere of the Jovian moon, TITAN. Trace 2 is the observed data with a full width at half maximum (FWHM) of 6 data points. Trace 3 is the deconvolved estimate of FWHM 2 data points.

In order to challenge the neural network in Fig. 1 a random spectrum (random positions and intensities) was synthesized. Actually two spectra were created, one of FWHM 6 and one of FWHM 2 -- (the Image and Object respectively in the parlance of Fig. 2). These were used to train the neural network of Fig, 1. Samples of the Image spectrum were presented to the neural network and an output (network out)generated. This network output was compared to the Object (true data) and the network parameterization adjusted in such a way that if the same piece of training data was presented immediately again, the sum square error would have decreased.

Training was carried out on a stochastic basis in an iterative fashion until the network had 'learned its lesson.' At that point, the actual diode laser spectrum of Fig. 3, trace 2 was presented to the network (parameterization having been frozen at completion of training) and the network produced the spectrum shown in trace 1 of Fig. 3. One notes that the results are far from perfect. On the other hand, the question asked was very demanding since the network had never "seen" the laser spectrum during training.

These results led us to believe that neural networks were of some consequence and that they seemed to be capable of significant generalization. Note that the black box as neural network was -- in a manner of speaking -- taught to deconvolve numerous statistical samples of the type of spectra shown in Fig. 3. Training consisted of presenting spectra like trace 2 in Fig. 3 to the input, generating an output, and comparing the output with the model (desired) output. From the difference between the actual and desired output, an error signal was generated and used to modify the internals of the neural network so as to minimize the error signal.

A neural network is more than a mystical black box. Actually, the neural networks of which we speak are simulations of models of the premier neural network, the human brain. As mentioned in the introduction, computational simulations of neural networks have their origins in brain models. Our interests, however, are in the utilization of neural computational systems (simulated neural networks) to solve real world problems. It is thus time to put the black boxes away -- they have served a purpose -- and to now address simulated neural networks in somewhat more detail.

There are many types of simulated neural networks. From here on, we shall presume that when we speak of a neural network that it is meant to be a numerically simulated neural network. A very good summary of the range of types of neural networks can be found in a review by Simpson [10].

Herein we shall restrict our discussion of the realization of artificial neural networks to **Feed Forward, Back Propagation Error Correction networks** -- a backprop net, for short. A schematic of the structure of a backprop net is shown in Fig. 4.

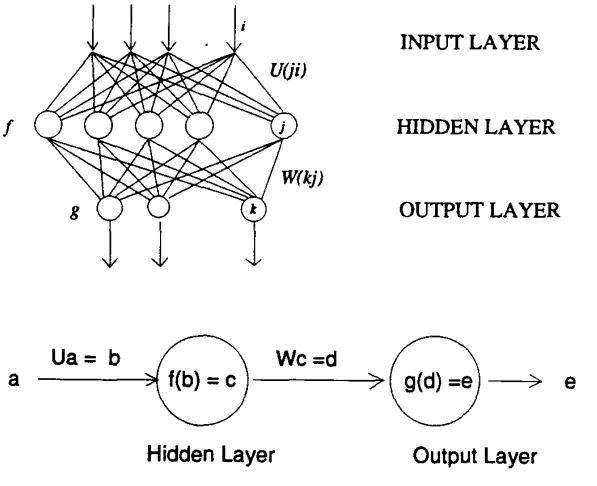

Fig. 4. A Schematic representation of a back-prop net. The small circles represent processing elements, sometimes called neurodes and these produce an output that is some general function of the sum of the inputs to the processing element. The input data is a which is called b as soon as it is in the net. b is processed by the weight function U and passed to the hidden layer processing elements as the vector c. Each processing element sums all of the inputs and generates an output d by operating on c. d is then presented to the output layer after processing by the output layer activation function g(). The network output is the vector f.

The activation functions that make up the functionality of the processing

elements are often sigmoidal functions that bear a great resemblance to hyperbolic tangent functions. A typical sigmoid function that makes up a processing element is given as

$$f() = (1/(1+\exp(\alpha a + \varphi)) \quad (2)$$

Which takes on a value of zero for a large positive, 1/2 for a zero, and one for a large negative. φ is considered a threshold and for practical purpose may be thought of a shifting the origin of the function in Eq.(2). On the other hand, α determines the effective slope of the transition region - with small values resulting in a nearly linear transition region. Large values result in a steep slope transition region and as such simulate a two-state function. Intermediate values of α result in a highly non linear processing element. Network behavior may be strongly dependent on the choice of slope factor and threshold. In a given layer of a network, the slope factor is often held fixed while the thresholds are varied along with the weights.

The back propagation network is fairly easy to describe analytically. Refering to Fig. 4, the network input **a** is defined as a vector

$$\mathbf{a} = (a_1, \ldots, a_N) \quad (3)$$

The input to the middle layer **b** is given by the product of the weight matrix **U** and the input vector **a**

$$\mathbf{b} = \mathbf{U}\mathbf{a} \quad (4)$$

Moving through the layer of feed forward neurodes or processing elements, the activation or processing function, such as the one in Eq.(2), is applied to each element in the vector **b**, as

$$\mathbf{c} = f(\mathbf{b}) \quad (5)$$

This complicates treating the process analytically since a new algebra would be required to treat the backprop net in compact form. To complete the transit of the network by an input vector **a**, the process is repeated by applying the weight matrix **W** to the vector **c** generating vector **d** that is processed by the activation function (cf. Eq.(2)) g() generating the network output **e** as

$$\mathbf{e} = g(\mathbf{W} \cdot f(\mathbf{U}\mathbf{a})) \quad (6)$$

Equation (6) is best understood by reference to Fig. 4.

Setting up and training a network is fairly routine -- once you have done it the first time. We shall use a non-trivial example to lead the reader through the process. The example is related to image enhancement (recovery) in the presence of spherical aberration and defocus such as is the situation with the Hubble Space Telescope (HST). The example deals with a simulation and uses a back--propagation error--correction feed--forward network (as in Fig. 4) to do the recovery. The point spread function (PSF) of HST is relatively complicated but is approximated by a combination of Gaussian functions as described in the Fig. 5 caption. Using such a PSF (in linear systems theory the PSF is the instrumental response function) a set of simulated star fields was set up using Gaussian stars with a fixed full width at half maximum (FWHM), random position, random number of stars (though limited to 5 in the 32x32 image space), and random brightness. The object star fields were convolved with the simulated HST PSF yielding a total of 24 corrupted images. We refer to each image used to train the network as a lesson. Of the 24 available lessons, three were reserved as controls to be used to test the quality of the trained network.

The 32x32 images and objects (to be used as net input and comparison output respectively) were treated as linear 1024 point linear brightness vectors. It is important to stress that this is of no consequence to the neural network since the same or equivalent mapping is achieved whether one treats the input as a linear vector or a 2-dimensional array.

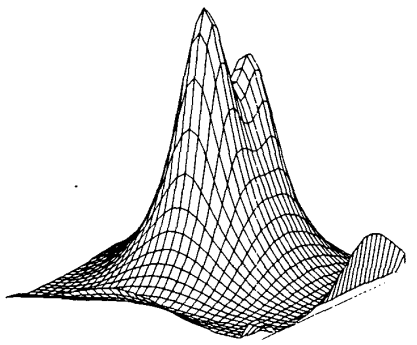

Fig. 5. Simulated star field in 3-dimensional display as might be observed by the Hubble Space Telescope. The star field is synthesized using Gaussian stars convolved with a Hubble-like point spread function (instrumental response function) with about 18% of the energy contained in a central, sigma = 1 Gaussian and 82% of the energy deployed in a Gaussian with sigma = 6. The training data set is fabricated with a random number of "stars" in its 32x32 image space, with each "star" having a randomly chosen intensity.

The same is true for the output -- which is treated in the neural network simulation as a linear, 1024 point brightness array related to the recovered image. (It is the recovered image, it is just arranged differently from the way we believe images should be arranged.)

The configuration of the network was found to be an important factor relative to successful training and generalization. In this test case we finally used a hidden layer dimension of 11 although successful cases used between 11 and 23 in the hidden layer (cf. Fig. 4).

Training of the network requires that the corrupted image be presented to the network and the signal propagated to the output (by the simulation program). The net output and the object (desired recovery) from the lesson are compared and an error signal derived. In this case the error signal is just the result of a simple gradient search correction generated for each weight between the hidden and output layer (cf. reference 10). These weights are corrected and the weights leading the signal to the hidden layer are corrected next. While the actual hidden layer weight correction is subtle and difficult to describe, it is easy to do. What one does is back propagate a correction to a given hidden layer weight by analytically allocating to each weight element a correction weighted by the output layer weights and summed over all contributors [10].

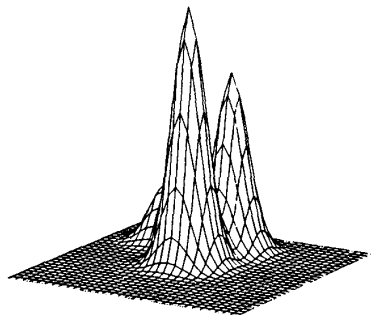

Fig. 6. The synthesized true object -- a star field composed of a random number of randomly positioned, random brightness Gaussian "stars." The "stars" have sigma = 2. These are the ground truth objects that the neural network attempts to recover from the network input shown in Fig. 5.

In Fig. 6 the desired output (actual simulated object) for one of the control images is presented. The trained network response to the input of the image in Fig. 5 is shown in Fig. 7. One sees that the network has learned its lesson quite well and has in fact been able to generalize successfully to some extent.

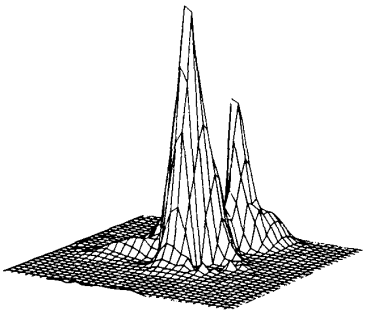

Fig. 7. The recovery estimate from the backprop network trained on 21 random "star" fields. In this case, 10,000 presentations of the 21 "lessons" were needed to train the network. As in the case depicted in Fig. 3, the network parameterization was frozen prior to presentation of the input shown in Fig. 5. Prior to the control lesson presentation, the network had never been presented the control data.

The actual training process required the random selection of 21 lessons per training step from the set of 21 candidates (the 3 reserved control lessons were not used in training). Weights were corrected after each lesson -- and not after each suite of lessons. This is necessary if successful generalization is to be achieved.

Using a priori knowledge to constrain the network has been found to be important where one desires to do the types of things suggested in this paper. We have found, in collaboration with Gordon Chin at Goddard Space Flight Center, that use of a priori knowledge about the physical system and problem being treated can improve the results significantly -- examples will be presented.

References

[1] D. E. Rumelhart and J. L. McClelland, **Parallel Distributed Processing: Explorations in the Microstructure of Cognition,** Vol. 1, MIT Press, Cambridge, MA, 1986.

[2] D. E. Rumelhart and J. L. McClelland, **Parallel Distributed Processing: Explorations in the Microstructure of Cognition,** Vol. 2, MIT Press, Cambridge, MA, 1986.

[4] R. P. Lippman, IEEE ASSP Magazine **4**, 4 (1987).

[5] Yoh-Han Pao, **Adaptive Pattern Recognition and Neural Networks**, Addison-Wesley, Reading, MA, 1989.

[6] **DARPA Neural Network Study**, AFCEA International Press, Fairfax, VA, 1988.

[7] W. E. Blass and G. W. Halsey, **Deconvolution of Absorption Spectra**, Academic Press, New York, 1981.

[8] W. E. Blass and G. W. Halsey, Chapter 6 in **Deconvolution,** ed. P. A. Jansson, Academic Press, New York, 1984.

[9] G. W. Halsey and W. E. Blass, Chapter 7 in **Deconvolution,** ed. P. A. Jansson, Academic Press, New York, 1984.

[10] Patrick K. Simpson, **Artificial Neural Systems, Foundations, Paradigms, Applications and Implementations,** Pergamon Press, New York, 1990.