COMMUNICATIONS AND NEURAL NETWORKS: THEORY AND PRACTICE

Mark D. Plumbley

Division of Engineering, King's College London, Strand, London WC2R 2LS, UK

ABSTRACT

In this paper we shall see that neural networks and communications are interlinked in a number of ways, towards the goal of efficient communication of information. One concrete example of this is the use of neural networks to ensure efficient use of communication channels, through connection admission control in ATM networks. In addition, however, efficient communication is also important within a decision making system such as a neural network. Finally we examine what type of neural network solutions are suggested by this approach.

1. INTRODUCTION

In this paper we will be concerned with how the application of neural networks is related to problems of efficient communication of information. For example, we shall see that neural networks can be used to aid decision making in ATM connection admission control, attempting to make most efficient use of an ATM network. In this example, the link between neural networks and communication efficiency is relatively direct.

However, a great many other problems can also be formulated in terms of efficient communication of information. Some of these are fairly obvious, as in the case of the ATM network above, while others may be a little more indirect. In these the *communication* concerned may be within the neural network or other decision-making system itself, and the information may be that which is required to make a later decision.

Consider a typical pattern recognition system $f(\mathbf{x})$ which is designed to classify an input $\mathbf{x}$ into one of a number of classes ω_i. A common approach (often called the *Bayesian* or *maximum a-posteriori* (MAP) approach) is to try to design a system which will choose an output class $\hat{\omega}_i$ which will maximize the probability $P(\omega_i|\mathbf{x})$. This may be done by attempting to model the likelihood $p(\mathbf{x}|\omega_i)$ and the a-priori probabilities $P(\omega_i)$ and using Bayes theorem to maximize $P(\omega_i|\mathbf{x})$.

In practice, however, the input $\mathbf{x}$ is often very large or complex (e.g. if $\mathbf{x}$ is an image) that it is not practical to construct these models directly, and a pre-processing stage is used. Thus the system $y = f(\mathbf{x})$ is split into $\mathbf{v} = f_1(\mathbf{x})$ and $y = f_2(\mathbf{v})$ where $f_1(\cdot)$ is a *pre-processing stage* on the front of the system. The purpose of this pre-processor is to simplify the representation, while retaining as much information as possible that would affect the final decision to be made by $f_2(\cdot)$.

We are therefore faced with a problem of efficient communication: how to construct $f_1(\cdot)$ (and $f_2(\cdot)$) to transmit as much of the information in $\mathbf{x}$ pertinent to the classification decision about ω_i as possible, while making the representation $\mathbf{v}$ used to transmit this information as simple as possible.

This same type of problem is also faced by biological organisms in attempting to make sense of the world around them. We shall see how the cross-fertilization of ideas from biology and communications are pointing to new directions for neural network applications.

2. NEURAL NETWORKS

There has been interest in the computational capabilities of biological neurons for at least the last 50 years. While neural networks which are applied to information engineering problems are *inspired* by our knowledge of biological neural processing, we do not normally try to model a particular biological system in detail. For example, biological neurons normally communicate their activity to each other through a series of all-or-nothing pulses (*spikes*), while we use either binary or real-valued numbers to represent activity in an artificial neural network. In a biological system, there are many different types of neurons specialized for particular tasks, while we normally use only one or two in types in a given neural network. We sometimes make the distinction between *artificial neural networks* (ANNs) and biological neural networks if we want to make this clear.

In neural network models used for problem solving, we start from the principle of performing a complex task with a network of many neurons, each of which performs a relatively simple function by itself. The information that the neural network 'knows' about a given problem is really held in the strengths of the connections between the neurons, rather than the neurons themselves. The ability to tackle a complex problem is then an emergent property of the neural network itself, rather than being a property of any individual neuron.

2.1. Neural Network Basics

As we mentioned above, (artificial) neural networks are composed of a number of neurons, or *nodes*. Each node has a number of *inputs* x_i connected to it via *weights* w_i, and calculates its output y as a simple function of its inputs.

Nodes may calculate different functions in different types of neural networks, but a typical node might calculate a *weighted sum* of the inputs,

$$s = \sum_{i=1}^{N} w_i x_i \tag{1}$$

and feed this through a *sigmoidal* (s-shaped) *activation function* such as to give the node output:

$$y = \sigma\left(\sum w_i x_i\right). \tag{2}$$

A number of other node functions are possible. For example, the sigmoid activation function could be replaced by a 0-or-1 *threshold* function, or the weighted sum may be replaced by a squared distance (with a suitable activation function) to give a *radial basis function* (RBF) node:

$$y = \exp\left(-\left(\sum (w_i - x_i)^2\right)\right). \tag{3}$$

Individual nodes (neurons) are connected together via their weights to form the neural network.

In a layered neural network, such as a *multi-layer perceptron* (MLP) the outputs of the neurons in one layer are fed forward to the inputs of the next layer. Any nodes which are not connected to either the input or the output of the network are called 'hidden' nodes. By adjusting the weights, such a neural network can be trained to perform a range of different input-output functions, in much the same way that adjusting the parameters a_i in a polynomial

$$y = a_o + a_1 x + a_2 x^2 + \cdots \tag{4}$$

will change the function the polynomial follows. The procedure used to adjust the weights is called the *learning algorithm*.

2.2. Supervised learning

Neural networks are often required to learn an input-output mapping from existing data. For example, suppose that have data such as income, age, and amount requested for a bank loan (input data), together with the yes/no decision that a skilled expert had previously made (desired output data, or *target*): this is called the *training data*. We can train the network by adjusting its weights until the mismatch between its outputs and the targets, often measured as the *mean-squared error* (MSE), is a small as possible over all the training data. We can then use the network by presenting it with the input data only, and the network output should closely match the target output for that input.

The most popular supervised learning algorithm for training a supervised network such as an MLP is *Error Back-Propagation*, usually called 'BackProp'. This is simply a mechanism that finds the derivative (i.e. the slope) of the MSE with respect to each of the weights in the network, and then uses a steepest-descent search to find the set of weights which gives the minimum error. Neural networks can take a long time to train, making many passes over their training data.

2.3. Other networks

There are several other types of neural network and learning algorithm, as well as those mentioned above. For example, Hopfield-type networks uses bi-directional connections and are used to tackle optimization problems, or as associative memory models.

Nodes in RAM-based networks such as the WISARD or pRAM (sometimes called n-tuple networks or 'weightless' networks) use binary inputs as the address lines to a small RAM inside the node. The node function is altered by changing the contents of the RAM. Some networks can learn to produce the correct output given only an indication of reward or penalty, like the fitness term used for genetic algorithms. This is called *reinforcement* learning.

Others, so-called *unsupervised* networks, are not given any guidance on the 'correctness' or otherwise of an output. We shall see later that some of these unsupervised networks can be formulated as a data compression task within our efficient communication framework.

For more details of these and other neural networks, the reader is referred to one of the many standard texts, such as that by Haykin [3].

3. NEURAL NETWORKS IN COMMUNICATIONS

An interesting concrete application of neural networks in communications is in the control of ATM (Asynchronous Transfer Mode) networks. In ATM networks, users send information in short fixed-sized *cells*, which are packets of 48 data bytes plus a 5 byte header. ATM network nodes contain hardware switches that route each packet according the information in the header. There is no synchronization between different users, so ATM nodes include output buffers to absorb fluctuations in the rate that cells from several sources arrive for one link. As the buffer fills up, the delay before cells is transmitted increases, and if the buffer fills completely some cells will be lost. (Note that communications engineers and neural network researchers both use the term *network* for different purposes, so beware of possible confusion!).

When a new users wishes to connect to a node, the node must decide whether or not to admit the connection. To bring in most revenue, the ATM network must carry as much traffic as possible, while keeping minimum levels of *quality of service* (QoS) for each of the users, i.e. limiting cell delay and cell loss. Predicting whether the new connection would degrade the QoS unacceptably a difficult problem, and is one where neural networks have been brought to bear [4].

Before the network is installed in the ATM switch, the network is trained to match either previously-collected training data (of cell arrival statistics vs. cell loss rate), or to match a model of the behaviour if a good model is available. Once trained, the neural network can be used in the connection admission controller to predict e.g. the cell loss rate that would result if a new connection is admitted. If this predicted loss rate is above a certain threshold, the new connection would be rejected. While the network is being used, more information is available concerning the real cell loss rates during operation of the ATM switch.

The neural network can be trained using this information, to adapt more closely to the traffic characteristics: this is called *real-time* training.

Other neural networks, such as the pRAM, have also been used in ATM networks [10]. For an overview and a sample of applications of neural networks to communication networks, see the recent special issue of IEEE Communications Magazine [5].

4. COMMUNICATIONS IN NEURAL NETWORKS

While we have seen that communications engineering is taking advantage of neural networks, it is also true that neural networks can also use ideas from communications engineering.

In a biological sensory system such as vision, the information from the world outside the organism needs to be processed into a form suitable for higher stages in the brain, using the biological neural networks in the organism. But this processing involves certain costs and constraints, such as the number of neurons which will be used, the space available, the noise in the system, and the energy used in processing the signal (the human brain uses some 20% of the body's energy intake just to process information!).

One proposal for learning in unsupervised neural networks, is that they should learn to maximize the Shannon information transmitted through them: Linsker's *Infomax* principle [7]. On the biological side, Atick and Redlich [2] have found a very good match between the measured response of the human visual system, and the response properties that the human retina should have predicted by this information-theoretic approach.

4.1. Principal subspace networks

In artificial unsupervised neural networks, we can construct algorithms which will learn to perform Infomax. Provided we make some simple assumptions about the input noise, in the linear 2-layer networks we mentioned earlier, Linsker shows that information capacity is maximized when the network performs principal component analysis (PCA).

Suppose we have an N-dimensional zero-mean input random vector $\mathbf{x}_\mathrm{a}$ and an M-dimensional output vector $\mathbf{y}_\mathrm{a} = [y_1, \ldots, y_M]$ where $\mathbf{y}_\mathrm{a} = \mathbf{W}\mathbf{x}_\mathrm{a}$ and $\mathbf{W}$ is an $M \times N$ weight matrix with $M < N$ (figure 1(a)). If we set the M successive rows of $\mathbf{W}$ to be the M largest eigenvectors of the input covariance matrix $\mathbf{\Sigma}_{\mathbf{x}_\mathrm{a}} = E(\mathbf{x}_\mathrm{a}\mathbf{x}_\mathrm{a}^T)$, then we have performed a principal component analysis of the input $\mathbf{x}_\mathrm{a}$.

If the input $\mathbf{x}_\mathrm{a}$ is corrupted by equal-variance additive gaussian noise, then PCA optimizes the information capacity from $\mathbf{x}_\mathrm{a}$ to $\mathbf{y}_\mathrm{a}$. If fact, for optimum information capacity it is sufficient for the rows of $\mathbf{W}$ to span the same subspace as the first M eigenvectors of $\mathbf{\Sigma}_{\mathbf{x}_\mathrm{a}}$: we call this principal *subspace* analysis.

A number of neural network algorithms have been developed to perform PCA or principal subspace analysis [14, 9, 12], many based on the Oja [8] principal component finding neuron. Often these use a learning algorithm of the form

$$\Delta\mathbf{W} = \eta_\mathbf{W}(\mathbf{y}_\mathrm{a}\mathbf{x}_\mathrm{a}^T - \mathbf{K}\mathbf{W}) \tag{5}$$

where $\eta_\mathbf{W}$ is a small update factor, and $\mathbf{K}$ is a weight decay matrix which is typically a function of $\mathbf{W}$ and $\mathbf{x}_\mathrm{a}$.

4.2. Decorrelation networks

Any practical signal processing system will be faced with having to cope with noise and inaccurate representation of data within it. Suppose we are now concerned with ensuring that our neural network should protect the outgoing signal against processing noise on its output. If simplifying assumptions can be made, it turns out that the network should learn to make its outputs as uncorrelated and equal-variance (i.e. *white*) as possible to optimize communication efficiency [11].

For a neural network and algorithm to achieve this, suppose that we have a network with an input vector $\mathbf{x}_\mathrm{b}$, output vector $\mathbf{y}_\mathrm{b}$, and interneuron vector $\mathbf{z}$, where these are all M-dimensional (figure 1(b)). We further suppose that $\mathbf{y}_\mathrm{b}$ and $\mathbf{z}$ adapt over a fast timescale so that

$$\mathbf{z} \leftarrow \eta_\mathbf{z}(\mathbf{V}^T\mathbf{y}_\mathrm{b}) + (1-\eta_\mathbf{z})\mathbf{z} \qquad \mathbf{y}_\mathrm{b} \leftarrow \eta_\mathbf{y}(\mathbf{x}_\mathrm{b} - \mathbf{V}\mathbf{z}) + (1-\eta_\mathbf{y})\mathbf{y}_\mathrm{b} \tag{6}$$

where $-\mathbf{V}$ is a matrix of inhibitory connections from $\mathbf{z}$ to $\mathbf{y}_\mathrm{b}$, which are equal and opposite to the connections $\mathbf{V}^T$ from $\mathbf{y}_\mathrm{b}$ to $\mathbf{z}$, and $\eta_\mathbf{y}$ and $\eta_\mathbf{z}$ are the update factors for $\mathbf{y}_\mathrm{b}$ and $\mathbf{z}$ respectively. The equilibrium activity is then given by

$$\mathbf{z} = \mathbf{V}^T\mathbf{y}_\mathrm{b} \qquad \mathbf{y}_\mathrm{b} = \mathbf{x}_\mathrm{b} - \mathbf{V}\mathbf{z} \tag{7}$$

so that we have

$$\mathbf{y}_\mathrm{b} = (\mathbf{I} + \mathbf{V}\mathbf{V}^T)^{-1}\mathbf{x}_\mathrm{b} \tag{8}$$

provided the term $(\mathbf{I} + \mathbf{V}\mathbf{V}^T)$ is positive definite. This settling is assumed to operate at a timescale much faster than any change in the weights $\mathbf{V}$.

In [11], it was shown that the simple local algorithm

$$\begin{aligned} \Delta\mathbf{V} &= \eta_\mathbf{V}(\mathbf{y}\mathbf{z}^T - \beta\mathbf{V}) \\ &= \eta_\mathbf{V}(\mathbf{y}\mathbf{y}^T - \beta\mathbf{I})\mathbf{V} \end{aligned} \tag{9}$$

where $\eta_\mathbf{V}$ is a small learning rate, produces the decorrelated equal variance outputs required, provided this can be achieved by *reducing*, and not increasing, the variance of all the input components.

The algorithms above can be combined to try to perform both the roles of signal extraction from input noise, and protection from output noise, towards our ultimate aim of efficient communication of information [13]. These combined networks include feedback inhibitory connections, which are thought to be important in many biological sensory and motor systems.

Investigating this type of artificial feedback network may help us to learn about biological information processing in these systems. The increasing interest in this area has led to the recent workshop series on *Information Theory and the Brain* (see [6] for the proceedings of the first of these). Also, since simple Hebbian-like learning algorithms can be used to find these optimal situations, this type of approach may be practical for the initial stages of an artificial vision system.

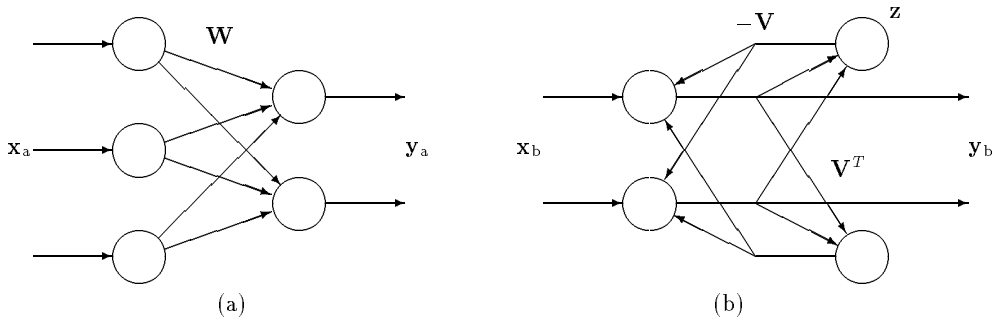

Figure 1. Linear networks for performing (a) principal subspace analysis, and (b) decorrelation.

4.3. Other possibilities

One interesting application from the use of principal components is the extraction of shape from shading in human heads by Atick, Griffin and Redlich [1]. Rather than attempting to solve the general shape-from-shading problem, they find that the variations between human heads are adequately represented by a very small PCA subspace of possible 3-D objects, simplifying the problem.

Here restricting the set of objects which the system must communicate about has allowed a simple (i.e. low-cost) intermediate representation to be used, achieving the goal of efficient communication within the system. With more work in this direction it may be that the technique of many relatively simple processing stages, that biological information processing systems seem to use, may begin to bear fruit in artificial information processing systems.

5. CONCLUSIONS

Neural networks are growing strong links with the communications field. We have seen not only that neural networks can be applied to applications in communication networks, but communication theory can also be applied to neural network learning. With neural networks becoming more and more a standard tool in the information engineer's armoury, and with continued advances in the theoretical background to neural networks, we could expect this trend to continue.

REFERENCES

[1] J. J. Atick, P. Griffin, and A. Redlich. Statistical approach to shape from shading: reconstruction of 3d face surfaces from single 2d images. *Neural Computation*, 1996. In press.

[2] J. J. Atick and A. N. Redlich. What does the retina know about natural scenes? *Neural Computation*, 4:196–210, 1992.

[3] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Macmillan, New York, 1994.

[4] A. Hiramatsu. ATM communications network control by neural networks. *IEEE Transactions on Neural Networks*, 1:122–130, 1990.

[5] IEEE Communications Magazine. Oct. 1995.

[6] ITB. Special issue on Stirling Workshop on 'Information Theory and the Brain'. *Network*, 7(2), 1996.

[7] R. Linsker. Self-organization in a perceptual network. *IEEE Computer*, 21(3):105–117, Mar. 1988.

[8] E. Oja. A simplified neuron model as a principal component analyser. *Journal of Mathematical Biology*, 15:267–273, 1982.

[9] E. Oja and J. Karhunen. On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix. *Journal of Mathematical Analysis and Applications*, 106:69–84, 1985.

[10] G. Onyiagha, X. Krasniqi, and T. G. Clarkson. Probabilistic RAM neural networks in ATM traffic shaping and policing. In *Proceedings of the Conference on Engineering Applications of Neural Networks, EANN'96*, pages 229–232, 1996.

[11] M. D. Plumbley. Efficient information transfer and anti-Hebbian neural networks. *Neural Networks*, 6:823–833, 1993.

[12] M. D. Plumbley. Lyapunov functions for convergence of principal component algorithms. *Neural Networks*, 8:11–23, 1995.

[13] M. D. Plumbley. Information processing in negative feedback neural networks. *Network*, 7(2):301–305, 1996.

[14] R. J. Williams. Feature discovery through error-correction learning. ICS Report 8501, University of California, San Diego, 1985.