Neural Networks: A Principled Perspective*

Christopher M. Bishop

Neural Computing Research Group
Dept. of Computer Science and Applied Mathematics
Aston University, Birmingham. B4 7ET, U.K.

C.M.Bishop@aston.ac.uk
http://www.ncrg.aston.ac.uk/

Technical Report: NCRG/95/014

Abstract

Introductory accounts of artificial neural networks often rely for motivation on analogies with models of information processing in biological networks. One limitation of such an approach is that it offers little guidance on how to find optimal algorithms, or how to verify the correct performance of neural network systems. A central goal of this paper is to draw attention to a quite different viewpoint in which neural networks are seen as algorithms for statistical pattern recognition based on a principled, i.e. theoretically well-founded, framework. We illustrate the concept of a principled viewpoint by considering a specific issue concerned with the interpretation of the outputs of a trained network. Finally, we discuss the relevance of such an approach to the issue of the validation and verification of neural network systems.

1 Introduction

There is a commonly held view that neural networks are unsuitable for use in high integrity or safety critical systems. That such a view should be widespread is unsurprising in view of the way in which neural networks are often presented. For example, introductory texts on neural networks frequently draw on historical analogies with information processing in biological systems to motivate network architectures and algorithms. While this lures the reader with the tantalising prospect of artificial systems with capabilities matching those of the human brain, it in fact offers little guidance on how to find optimal algorithms, let alone how to verify the correct performance of systems containing neural networks.

*Published in Proceedings ERA Technology conference *Neural Networks – Producing Dependable Systems*, November 1995, 3.1.1–3.1.9.

The problem is further exacerbated by the widespread availability of neural network software which allows the user to train network models and even create applications without the least idea of the theoretical basis for the models being used. The user manuals supplied with software packages almost invariably focus on the mechanics of using the software itself and rarely provide the reader with any insight into how to make effective use the algorithms[1].

A key goal of this paper is to draw attention to a quite different viewpoint in which neural networks are seen as algorithms for statistical pattern recognition based on a *principled*, i.e. theoretically well-founded, framework. As such neural networks extend and complement the many existing techniques for pattern recognition, and hence build on, rather than ignore, the substantial body of knowledge in this field accumulated over several decades. Indeed, it is only by the adoption of a principled viewpoint that neural networks can be used successfully to tackle non-trivial problems and that issues associated with the validation and verification of systems containing neural networks can be addressed.

In Section 2 we illustrate the concept of a principled viewpoint by considering a specific issue concerned with the interpretation of the outputs of a trained network. We show how a principled approach leads to significant insight into the motivation for neural network algorithms and how it can have a considerable impact on practical applications. Its relevance to the validation and verification problem is discussed in Section 3.

2 Example: Interpretation of Network Predictions

One of the most widespread applications for neural networks is to the task of classifying a set of input variables, described by a vector $\mathbf{x}$, into one of K classes which we shall denote by $\mathcal{C}_k$, where $k = 1, \ldots, K$. Typically the network model, for example a multi-layer perceptron, has one output corresponding to each of the classes, and is trained by minimizing an error function defined over a set of training data. There are many important and subtle issues which this raises, but we shall focus on just one, namely the meaning which should be ascribed to the outputs of the trained network when it is presented with new data. Even here we shall only touch upon some of the relevant issues. Our goal is not to provide a comprehensive exposition on this topic, but rather to illustrate the meaning of a principled view of neural networks and the advantages that it conveys. A more extensive and complete discussion of this topic can be found in Bishop (1995).

One common approach to the classification problem is to use a network with sigmoidal output units of the form

$$g(a) = \frac{1}{1 + \exp(-a)} \tag{1}$$

[1] For example, the manuals for one major commercial system for simulating neural networks, amounting to over 800 pages, contain extensive discussions of file formats and menu options, and yet nowhere in these could I find any discussion of the topic of generalisation!

and to minimize a sum-of-squares error function. Once the network is trained, new patterns are classified according to which of the network outputs has the largest value. Sometimes various heuristics might be employed to 'improve' the performance of the network. For example, the network might be given extra training on those patterns in the training set which it mis-classifies, in an attempt to reach 100% correct classification. As we shall see, such ad-hoc methods can be highly counter-productive.

2.1 Posterior probabilities

The simple application of a neural network described above is equivalent to using the network as a form of *non-linear discriminant*, in other words the network itself is used to make the decision on how to classify a new input. There is, however, a much more powerful interpretation of the use of neural networks in the context of classification problems, which we now discuss.

The key is to distinguish between the two distinct stages in the classification process, namely *inference* and *decision*. At the inference stage the goal is to determine the *posterior probabilities*, denoted by $P(\mathcal{C}_k|\mathbf{x})$, for the input vector $\mathbf{x}$ to belong to each of the classes $\mathcal{C}_k$. These probabilities can subsequently be used to make *decisions*, such as assigning the input vectors to classes. The role of the neural network model is to predict the probabilities, with the subsequent decision-making process being performed separately. We shall see in Section 2.2 how to arrange for the network outputs to represent the probabilities $P(\mathcal{C}_k|\mathbf{x})$. Here we consider the benefits of a probabilistic interpretation of the network outputs.

By arranging for the network outputs to approximate posterior probabilities we can exploit a number of results, many of which are not available if the network is used simply as a non-linear discriminant (Richard and Lippmann, 1991). These include:

1. **Minimum error-rate decisions**

   Except for relatively trivial problems, it will be the case that the probability density functions $p(\mathbf{x}|\mathcal{C}_k)$ for data belonging to each of the classes will overlap. This means that perfect classification of data is fundamentally impossible. However, we can seek a classification which is optimal according to some criterion. For instance, we can seek to minimise the probability of a new input being misclassified. Statistical theory (Duda and Hart, 1973) shows that this is achieved by assigning new patterns to the class for which the posterior probability is largest. It is important to note that, as a consequence of the class overlap, these probabilities need not be close to 0 or 1. Heuristic procedures, such as applying extra training using those patterns which fail to generate outputs close to the target values, will be counterproductive, since this alters the effective distributions of the training data and makes it *less* likely that the network will generate the correct probabilities.

2. **Compensating for different prior probabilities**

The posterior probabilities $P(\mathcal{C}_k|\mathbf{x})$ can be related to the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ through Bayes' theorem in the form

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{p(\mathbf{x})} \tag{2}$$

where $P(\mathcal{C}_k)$ are the *prior* probabilities (in other words the overall probabilities for observing patterns from the different classes). The denominator in (2) plays the role of a normalising factor ensuring that the posterior probabilities sum to one $\sum_k P(\mathcal{C}_k|\mathbf{x}) = 1$. It can be related to the quantities in the numerator using

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) \tag{3}$$

Sometimes the prior probabilities expected when the network is in use differ from those represented by the training set. It is then it is a simple matter to use Bayes' theorem (2) to make the necessary corrections to the network outputs. This is achieved simply by dividing the network outputs by the prior probabilities corresponding to the training set, multiplying them by the new prior probabilities, and then normalizing the results. Changes in the prior probabilities can therefore be accommodated without re-training the network. The prior probabilities for the training set may be estimated simply by evaluating the fraction of the training set data points in each class. Prior probabilities corresponding to the network's operating environment can often be obtained very straightforwardly since only the class labels are needed and no input data is required. As an example, consider the problem of classifying medical images into 'normal' and 'tumour'. When used for screening purposes, we would expect the prior probability of 'tumour' to be very small. To obtain a good variety of tumour images in the training set would therefore require huge numbers of training examples. An alternative is to increase artificially the proportion of tumour images in the training set, and then to compensate for the different priors on the test data as described above. The prior probabilities for tumours in the general population can be obtained from medical statistics, without having to collect the corresponding images. Correction of the network outputs is then a simple matter of multiplication and division.

3. Combining the outputs of several networks

Rather than using a single network to solve a complete problem, there is often benefit in breaking the problem down into smaller parts and treating each part with a separate network. By dividing the network outputs by the prior probabilities used during training, the network outputs become likelihoods scaled by the unconditional density of the input vectors. These scaled likelihoods can be multiplied together on the assumption that the input vectors for the various networks are independent. Since the scaling factor is independent of class, a classifier based on the product of scaled likelihoods will give the same results as one based on the true likelihoods. This approach

has been successfully applied to problems in speech recognition (Bourlard and Morgan, 1990; Singer and Lippmann, 1992).

4. **Minimum risk**

   The goal of a classification system may not always be to minimize the probability of misclassification. Different misclassifications may carry different penalties, and we may wish to minimize the overall *loss* or *risk*. Again the medical screening application provides a good example. It may be far more serious to mis-classify a tumour image as normal than to mis-classify a normal image as that of a tumour (since the latter may lead to wasted effort in conducting more detailed tests, while the former may result in the patient's death). In this case, the posterior probabilities from the network can be combined with a suitable matrix of loss coefficients to allow the minimum-risk decision to be made. Again, no network re-training is required to achieve this.

5. **Rejection thresholds**

   In general we expect most of the misclassification errors to occur in those regions of $\mathbf{x}$-space where the largest of the posterior probabilities is relatively low, since there is then a strong overlap between different classes. In some applications it may be better not to make a classification decision in such cases. This is sometimes called the *reject option*. For the medical classification problem, for example, it may be better not to rely on an automatic classification system in doubtful cases, but to have these classified instead by a human expert. We then arrive at the following procedure

$$\text{if} \quad \max_k P(\mathcal{C}_k|\mathbf{x}) \quad \begin{cases} \geq \theta, & \text{then classify } \mathbf{x} \\ < \theta, & \text{then reject } \mathbf{x} \end{cases} \tag{4}$$

   where θ is a threshold in the range $(0, 1)$. The larger the value of θ, the fewer points will be classified. One way in which the reject option can be used is to design a relatively simple but fast classifier system to cover the bulk of the feature space, while leaving the remaining regions to a more sophisticated system which might be relatively slow. The reject option can be applied to neural networks provided the outputs of the network represent the posterior probabilities.

2.2 Error functions and activation functions

Having recognised that it is desirable for the network outputs to represent probabilities, we now have to face the problem of how to arrange for this to occur. There are many different choices of error function, activation function, target data coding and so on, all of which would be expected to alter the predictions made by the trained network. How are we to decide on the appropriate choices to achieve the desired goal?

Consider first a very simple problem of a one-dimensional input space and two classes, each of which is described by a Gaussian probability distribution with variance σ^2, so that

$$p(x|\mathcal{C}_k) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{(x-\mu_k)^2}{2\sigma^2}\right\} \tag{5}$$

We can use Bayes' theorem to find an expression for the probability of an input x belonging to class 1, so that

$$\begin{aligned} P(\mathcal{C}_1|x) &= \frac{p(x|\mathcal{C}_1)P(\mathcal{C}_1)}{p(x|\mathcal{C}_1)P(\mathcal{C}_1) + p(x|\mathcal{C}_2)P(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} \end{aligned} \tag{6}$$

where we have made use of (5). Here the quantity a is given by $a = wx + w_0$ with

$$w = \frac{(\mu_1 - \mu_2)}{\sigma^2} \tag{7}$$

$$w_0 = -\frac{(\mu_1^2 - \mu_2^2)}{2\sigma^2} - \ln\frac{P(\mathcal{C}_2)}{P(\mathcal{C}_1)} \tag{8}$$

Thus the posterior probability is given by a single-layer network with a sigmoidal activation function at the output unit. This analysis can be extended to non-linear network models and to a very general class of distributions called the *exponential family* (Bishop, 1995). The result is that the network output should again make use of a logistic sigmoid activation function. In this case the network has a single output whose value gives $y = P(\mathcal{C}_1|x)$, with the corresponding probability for class $\mathcal{C}_2$ given by $1 - y$. The correct choice of error function can be obtained from the principle of maximum likelihood, and takes the form

$$E = -\sum_{n=1}^{N} \{t^n \ln y^n + (1-t^n)\ln(1-y^n)\} \tag{9}$$

where n labels the training patterns.

For more that two classes, there will be one output per class, but the correct activation function is not a separate sigmoid for each unit (as is commonly seen in the neural networks literature) but a *softmax* or *normalized exponential* function of the form

$$y_k = \frac{\exp(a_k)}{\sum_{k'} \exp(a_{k'})} \tag{10}$$

where a_k represents the total summed input to the kth output unit. In this case the appropriate error function is given by

$$E = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_k^n \ln y_k^n \tag{11}$$

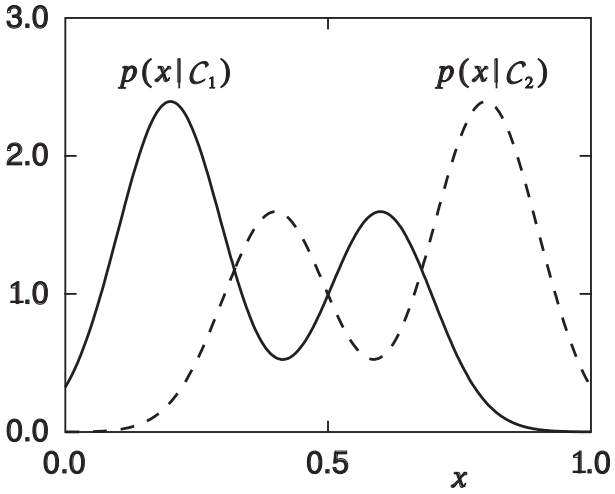

Figure 1: Plots of the class-conditional densities used to generate a data set to demonstrate the interpretation of network outputs as posterior probabilities. A total of 2000 data points were generated from these densities, using equal prior probabilities.

Clearly the error functions (9) and (11) have quite different forms from the standard sum-of-squares error.

Note that the average of each network output over all patterns in the training set should approximate the corresponding prior class probabilities, since

$$P(\mathcal{C}_k) = \int P(\mathcal{C}_k|\mathbf{x})p(\mathbf{x})\,d\mathbf{x} \simeq \frac{1}{N}\sum_{n=1}^{N} P(\mathcal{C}_k|\mathbf{x}^n) \tag{12}$$

These estimated priors can be compared with the sample estimates of the priors obtained from the fractions of patterns in each class within the training data set to provide an indication of the extent to which the actual network outputs are close to the required probabilities (Richard and Lippmann, 1991).

As a simple illustration, consider again a one-dimensional input space, and two classes whose distributions are shown in Figure 1. A data set generated from these distributions was used to train a multi-layer perceptron network. The resulting network function is illustrated in Figure 2 along with the true posterior probability calculated from Bayes' theorem.

3 Discussion

We have considered one specific aspect of neural networks, namely the interpretation of the network outputs for classification problems, as an illustration of the concept of a principled perspective. The same approach can be applied throughout the field of neural computing, and brings innumerable benefits including improved performance in applications, the avoidance of major pitfalls, and the ability to quantify many aspects of network and system performance. Such

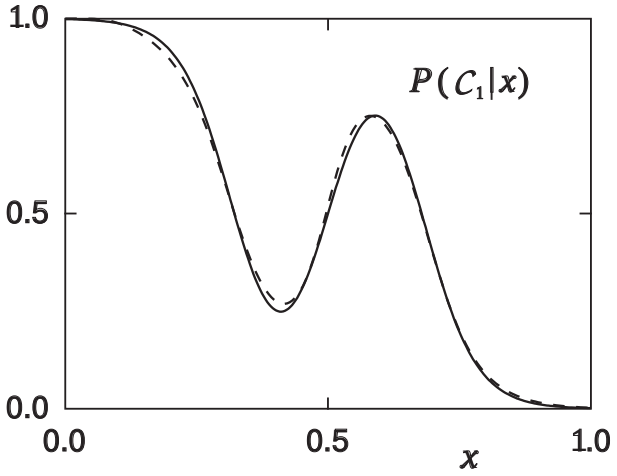

Figure 2: The result of training a multi-layer perceptron on data generated from the density functions in Figure 1. The solid curve shows the output of the trained network as a function of the input variable x, while the dashed curve shows the true posterior probability $P(\mathcal{C}_1|x)$ calculated from the class-conditional densities using Bayes' theorem.

issues are particularly relevant in the context of the validation and verification of systems containing neural networks.

Consider, for example, the problem of determining how reliable the predictions of a network are. Effective assessment of the uncertainty of network predictions requires a clear understanding of many effects such as the role of the input data distribution (Bishop, 1994b), the effects of uncertainty in the network parameters (Williams *et al.*, 1995), and the contributions from the intrinsic noise on the target data (Bishop, 1994a).

The adoption of theoretically well-founded view of neural networks is essential both to counter the hype and mistrust sometimes associated with these techniques, and to allow the effective application of neural networks in domains where the performance of the system is deemed critical.

References

Bishop, C. M. (1994a). Mixture density networks. Technical Report NCRG/94/001, Neural Computing Research Group, Aston University, Birmingham, UK.

Bishop, C. M. (1994b). Novelty detection and neural network validation. *IEE Proceedings: Vision, Image and Signal Processing* **141** (4), 217–222. Special issue on applications of neural networks.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press.

Bourlard, H. and N. Morgan (1990). A continuous speech recognition system embedding MLP into HMM. In D. S. Touretzky (Ed.), *Advances in Neural*

Information Processing Systems, Volume 2, pp. 186–193. San Mateo, CA: Morgan Kaufmann.

Duda, R. O. and P. E. Hart (1973). *Pattern Classification and Scene Analysis*. New York: John Wiley.

Richard, M. D. and R. P. Lippmann (1991). Neural network classifiers estimate Bayesian a-posteriori probabilities. *Neural Computation* **3** (4), 461–483.

Singer, E. and R. P. Lippmann (1992). Improved hidden Markov model speech recognition using radial basis function networks. In J. E. Moody, S. J. Hanson, and R. P. Lippmann (Eds.), *Advances in Neural Information Processing Systems*, Volume 4, pp. 159–166. San Mateo, CA: Morgan Kaufmann.

Williams, C. K. I., C. Qazaz, C. M. Bishop, and H. Zhu (1995). On the relationship between Bayesian error bars and the input data density. In *Proceedings Fourth IEE International Conference on Artificial Neural Networks*, pp. 160–165. Cambridge, UK: IEE.