Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition

JOHN S. BRIDLE
Speech Research Unit
Royal Signals and Radar Establishment
St. Andrews Road Great Malvern
UK WR14 3PS

Abstract

We are concerned with feed-forward non-linear networks (multi-layer perceptrons, or MLPs) with multiple outputs. We wish to treat the outputs of the network as probabilities of alternatives (*e.g.* pattern classes), conditioned on the inputs. We look for appropriate output non-linearities and for appropriate criteria for adaptation of the parameters of the network (*e.g.* weights). We explain two modifications: probability scoring, which is an alternative to squared error minimisation, and a normalised exponential (**softmax**) multi-input generalisation of the logistic non-linearity. The two modifications together result in quite simple arithmetic, and hardware implementation is not difficult either. The use of **radial units** (squared distance instead of dot product) immediately before the **softmax** output stage produces a network which computes posterior distributions over class labels based on an assumption of Gaussian within-class distributions. However the training, which uses cross-class information, can result in better performance at class discrimination than the usual within-class training method, unless the within-class distribution assumptions are actually correct.

1 Networks and Probabilistic Models

1.1 The stochastic model paradigm

Currently the most successful approach to automatic speech recognition is based on *Stochastic Models.* The approach generalises parametric statistical pattern recognition, and is also important in other application domains, such as computational vision.

NATO ASI Series, Vol. F 68
Neurocomputing
Edited by F. Fogelman Soulié and J. Hérault
© Springer-Verlag Berlin Heidelberg 1990

The method consists in treating the data as if it were the output of a stochastic system (one governed by probabilistic laws) [1]. The structure of the model is a pattern of direct dependencies between internal (hidden) and external (visible) random variables, and it is usually informed by insight into the structure of the data or of the real generator (*e.g.* vocal tracts in the case of speech). There is also a (possibly large) set of parameters of the model, which need to be estimated (such as conditional probabilities, means and variances). There are usually separate algorithms for recognition (interpreting the data in terms of the model) and for learning (estimating parameters.)

The basis of the recognition process is often to compute the likelihood of the data conditioned on some unseen random variable of interest, such as the class of the pattern: $P(\text{Data} \mid \text{Class})$. Using Bayes rule, the class conditional probabilities are then

$$P(\text{Class} \mid \text{Data}) = P(\text{Data} \mid \text{Class})P(\text{Class}) \Big/ P(\text{Data}),$$

where $P(\text{Class})$ are the prior probabilities (often assumed equal) and

$$P(\text{Data}) = \sum_{\text{classes}} P(\text{Data} \mid \text{Class})P(\text{Class}).$$

If we only require the most likely class then this denominator can be ignored.

One of the simplest stochastic models (and one we shall return to below) is a set of Gaussian distributions, one for each class. In this case the parameters are the priors, the means, and the covariance matrices. We may choose to restrict the covariance matrices in various ways, such as zero except on the diagonals (independence assumption), or still more by assuming the covariance matrix is unit diagonal times a constant. We might also assume that the covariance matrices for the different classes are related (*e.g.* equal). We would usually estimate the means etc. from the sample means, etc. (*i.e.* maximum likelihood estimation for each class separately).

1.2 Feed-forward nonlinear "neural" networks

In contrast to the stochastic model approach, a neural network approach typically starts with assumptions about the form of a suitable recognition process, such as a non-linear feedforward network, and we adjust its parameters (*e.g.* weights) to optimise some measure of performance.

The standard "multi-layer perceptron" (MLP) can be taken as a semi-linear logistic feed-forward network with squared error minimisation, trained using some optimisation technique exploiting the error back-propagation method for computing partial derivatives with respect to the weights. Such MLPs are often used for pattern classification, using one-from-N coding at the output. (*i.e.* One output for each class, targets are zero except for unity at the true class line.) However, this structure is really only suitable for cases where the outputs correspond to (approximately) independent properties of the input, such presence/absence of objects which may appear together, or in the use of componential representations, *e.g.* [2]. Later we consider a more appropriate structure for 1-from-N classification problems.

The usual criterion for adjusting the parameters (θ) of a network is minimum squared error between outputs and targets (y).

$$\min_{\theta} \mathrm{E}\left[(Y - F(x,\theta))^2\right]$$

The structure of the network may not allow this optimum to be reached, and if it does the search method may not find it. What we *should* try to optimise is the expected performance on *unseen* samples (but from the same distribution as the training data). In practice we minimise not the expected value of the squared error, but the average over the training set. Optimisation of a squared error criterion can tend towards networks producing estimates of the posterior distribution of classes. Later we consider an alternative score which is concerned with probabilities of correct classification decisions.

2 Stochastic models with neural network solutions

A natural way to think of the function of a neural-style network when used for pattern classification is in terms of conditional relative class density estimation, which is equivalent to the probability distribution over class labels conditioned on the input (i.e. the posterior distribution over class labels, $P(\text{Class} \mid \text{Data})$). The relative densities may need less parameters than the set of individual class densities, and the relative densities are precisely what matter in discrimination tasks.

To illustrate the connections between networks and stochastic models, we consider two simple stochastic models for which the relative densities are a semi-linear logistic funtion of the inputs.

2.1 Boolean random variables separately dependent on another BRV

As a simple example of a stochastic model, consider a set of Boolean random variables (BRVs) $\{X_i\}$ which are independent apart from their dependence on another BRV Y. The $(2n+1)$ parameters are $p_y \triangleq P(Y=1)$, $p_i \triangleq P(X_i=1 \mid Y=1)$ and $\tilde{p}_i \triangleq P(X_i=1 \mid Y=0)$. Suppose we are given observed values of the Xs, $x_1, \ldots, x_n$. and we wish to estimate the value of Y.

It can be shown that

$$P(Y=1 \mid x_1, \ldots, x_n) = 1/(1 + \exp -(w_0 + \sum_i w_i x_i)),$$

where

$$w_i = \log \frac{p_i}{(1-p_i)} - \log \frac{\tilde{p}_i}{(1-\tilde{p}_i)}, \quad \text{and} \quad \mathrm{w}_0 = \log \frac{\mathrm{p}_y}{(1-\mathrm{p}_y)} + \sum_i \log \frac{(1-\mathrm{p}_i)}{(1-\tilde{\mathrm{p}}_i)}.$$

Thus the semi-linear logistic is just what is needed in this case, to compute the probability of the unseen BRV, Y, being 1, as a function of the observed values of the Xs.

2.2 Equal covariance Gaussians

Consider next the case of discrimination between two classes, each univariate Gaussian distributed, differing only in the values of their means (m_a, m_b) and priors$(p_a \triangleq P(\text{Class}=a), p_b \triangleq P(\text{Class}=b))$ (*i.e.* equal variances, σ^2). The two probability density functions have the form

$$P(x \mid a) \triangleq P(X=x \mid \text{Class}=a) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m_a)^2}{2\sigma^2}}.$$

The posterior probability of class a, given an observation x, is

$$P(\text{Class}=a \mid X=x) = 1 \Big/ (1 + \exp -(w_0 + w_1 x)),$$

where

$$w_0 = \log(p_a/p_b) - \frac{(m_a^2 - m_b^2)}{2\sigma^2}, \qquad w_1 = \frac{(m_a - m_b)}{\sigma^2}.$$

Thus a single semi-linear logistic unit can compute the posterior distribution of class labels. This result extends to multi-variate distributions with equal covariance matrices.

Multi-layer networks are useful when the relationships between input and output are more complicated. In the following sections we are concerned only with the form of the output non-linearity and the criterion which is optimised by adjusting the weights. We treat first the case of independent outputs, which is relatively well established, then the case of present interest, which is a one-from-N classifier.

3 Networks with multi-component, independent outputs

If each output value, O_j, from a network is to be treated as a probability that an associated random variable, Y_j, is true, then we require that all N outputs lie between 0 and 1.

Given a set of unconstrained scalar input values, I_j, which may be weighted sums of outputs from other units $(I_j = \sum_i w_{ij} O_i)$ we wish to produce outputs of the network, $O_j = F(I_j)$ which satisfy the $0 > O_j > 1$ constraint. The term sigmoid is used for any non-decreasing function which maps the real line to the unit interval. The most commonly used sigmoid function is the logistic or (offset) hyperbolic tangent:

$$O_j = F(I_j) = \frac{1}{1 + e^{-I_j}} = \frac{1}{2}\left(1 + \tanh\frac{x}{2}\right).$$

One way to score such a system against the true values, $\{T_j\}$, is to say we shall use the Os to guess the true values. We shall guess that $Y_j = 1$ with probability O_j. If $T_j = 1$ we shall be correct O_j of the time, and if $T_j = 0$ we shall be correct $(1 - O_j)$ of the time. The probability of all the (independent) guesses being correct is

$$\prod_{j=1}^{N} O_j^{T_j}(1 - O_j)^{(1-T_j)}.$$

This is also appropriate if the Ts are between 0 and 1, signifying the *probability* that the random variable is true.

In fact we minimise minus the log of this product:

$$J = -\sum_j (T_j \log(O_j) + (1 - T_j)\log(1 - O_j)).$$

To compute the gradient of J wrt parameters of the network (*e.g.* weights) we need

$$\frac{\partial J}{\partial I_j} = \frac{\partial J}{\partial O_j}\frac{\partial O_j}{\partial I_j} = \frac{O_j - T_j}{O_j(1-O_j)} O_j(1 - O_j) = (O_j - T_j).$$

The difference between this partial derivative and the usual squared-error derivative is simply the removal of the $O(1 - O)$ factor.

This type of probability or entropy score for backpropagation networks has been suggested several times in different forms [3, 4, 5, 6]. It is essentially the same as the G measure used in the Boltzmann machine [7] and the recent Boltzmann Perceptron Network [8], which is a deterministic feedforward network closely related to the system presented in the next section. To our knowledge it has not yet been shown convincingly that such a score provides better performance than squared error in practice, but it certainly seems more appropriate for some purposes from a theoretical point of view. The closely related Maximum Mutual Information estimation method has been shown to improve results of hidden Markov model based speech recognition [9].

4 1-from-N Classifier Networks

For 1-from-N classifiers the *set* of outputs should have the form of a probability distribution over class labels, conditioned on the input:

$$O_j = P(c{=}j \,|\, \text{input}),$$

so we require

$$O_j > 0 \quad \forall j \quad \text{and} \quad \sum_{j=1}^{N} O_j = 1.$$

To enforce this stochastic constraint, we suggest a *normalised exponential* output non-linearity,

$$O_j = \mathrm{e}^{I_j} \Big/ \sum_k \mathrm{e}^{I_k}.$$

This **"softmax"** function is a generalisation of the logistic to multiple inputs. It also generalises maximum picking, or "Winner-Take-All", in the sense that that the outputs change smoothly, and equal inputs produce equal outputs. Although it looks rather cumbersome, and perhaps not really in the spirit of neural networks, those familiar with

Markov random fields or statistical mechanics will know that it has convenient mathematical properties. Circuit designers will enjoy the simple transistor circuit which implements it.

Softmax can approach arbitrarily close to a WTA effect by increasing an optional gain factor, $1/T$ (now use $e^{I_j/T}$. As $T \to 0$ the outputs approach a Kronecker delta function indicating the position of the maximum in the inputs. We have found it useful to consider replacing WTA with **softmax** in various algorithms. The result is often insight or an improved algorithm. It is easy to add an extra output with a constant input, to deal with rejects (inputs not in any of the main classes).

For scoring we assume that we shall pick one of the labels $(1, \ldots, N)$ based on the outputs treated as a distribution. (Pick j with probability O_j.)

First let us assume that for each input training pattern we have the true probability distribution over labels, $T_j = P(c = j \mid x)$. We wish our set of outputs, $\{O_j\}$ to be as close as possible to the true probabilities. An appropriate measure of similarity of the two probability distributions is the relative entropy of the Ts with respect to the outputs [10] (*i.e.* the extra information given by the T when we already know the Os)

$$G \triangleq -\sum_j T_j \log \frac{T_j}{O_j} = -\sum_j T_j \log T_j + \sum_j T_j \log O_j = H(T) - J,$$

where $J = -\sum_j T_j \log O_j$.

The first term is the entropy of the T distribution, and is independent of the outputs. The second term is the log of the probability of a long run of correct guesses, if we choose from the O distribution. If the true class, c, is known, the probability of choosing the correct class is O_c, and the expression for relative entropy simplifies because $H(T) = 0$ and $J = -\log O_c$.

In back-propagating the partial derivatives we note that each output depends on *all* the inputs to **softmax**, so:

$$\frac{\partial J}{\partial I_j} = \sum_k \frac{\partial J}{\partial O_k} \frac{\partial O_k}{\partial I_j}.$$

$$\frac{\partial J}{\partial O_k} = -\frac{T_k}{O_k}.$$

For the **softmax** output nonlinearity,

$$\frac{\partial O_k}{\partial I_j} = O_k(\delta_{kj} - O_j),$$

so

$$\frac{\partial J}{\partial I_j} = (O_j - T_j).$$

We conclude that **softmax** and the entropy score suit one another well.

5 Transistor circuits

It is well known that a long-tailed pair (differential pair) of transistors, each with an exponential transconductance characteristic (*e.g.* bipolar transistors or sub-threshold CMOS) has a logistic relationship between (differential) input voltage and output current. (In an amplifier we use the central, linear, portion of this transfer characteristic, and in a logic gate (*e.g.* ECL) we use the endpoints.) The **softmax** function can be implemented in a generalisation of the longtail pair, with one transistor per input value.

6 A Practical Example

To illustrate the use of the softmax nonlinearity, probability scoring and relationships between networks and conventional classifiers, we constructed a ("one layer") network which computes the posterior distribution over class labels under the following assumptions: the within-class distributions are all Gaussian, with equal covariance matrices (all the unit matrix times a known constant) and equal priors.

The outputs of the network were:

$$O_j = \mathrm{e}^{I_j} / \sum_k \mathrm{e}^{I_k},$$

where the inputs to the **softmax** were

$$I_j = -\sum_i (x_i - m_{ij})^2.$$

The components, m_{ij}, of the means of the distributions were the only free parameters, and they were adjusted using a back-propagation-based method, starting from the values of the within-class data means:

$$\frac{\partial J}{\partial m_{ij}} = \frac{\partial J}{\partial I_j}\frac{\partial I_j}{\partial m_{ij}} = 2(0_j - T_j)(x_i - m_{ij}).$$

The data was a set of first and second formant frequency values for spoken vowel sounds, in 11 vowel classes, spoken by many male and female American speakers [11][1].

The standard parameter estimation technique is maximum likelihood for each class separately. With the above assumptions about the form of the distributions, the parameters are the class means, and the ML estimates are the data means. The Bayes decision regions are shown in figure 1 (the steps in the lines are an artifact). Each point on the plane is labelled with the class of the nearest mean. Figure 1 also shows correctly classified training points as dots, and wrongly classified training points as letters corresponding to true classes.

[1]We are grateful to Richard Lippmann of Lincoln Laboratories for his version of this data [12]

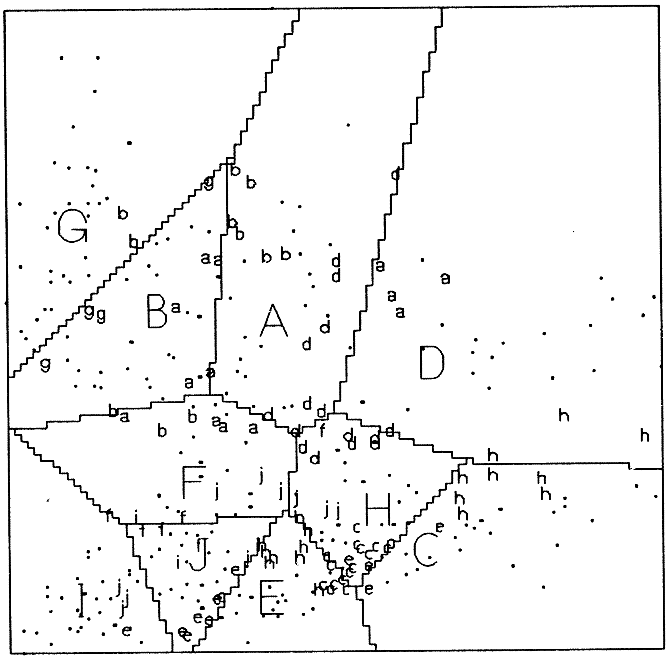

Figure 1. Decision boundaries etc using data means

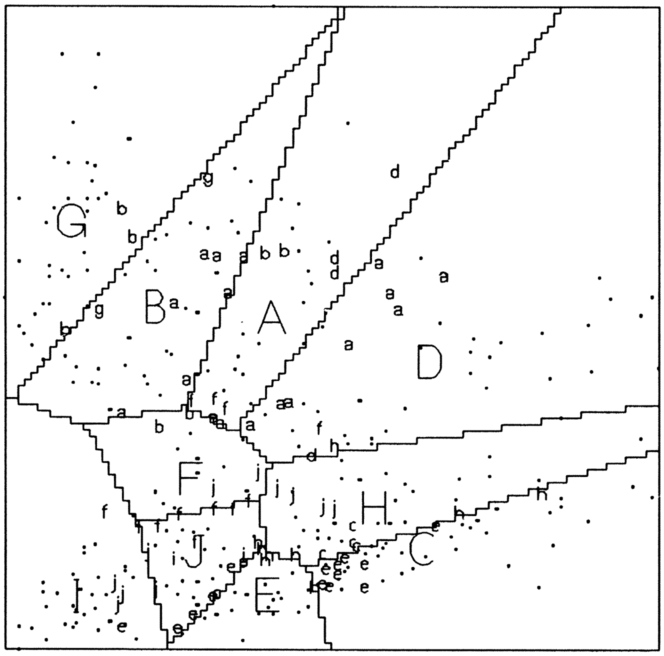

Figure 2. Decision boundaries etc after network training

68% of a separate test set was correctly classified by this "Euclidean distance to class mean" rule. When the reference points were adjusted further using the network method, the performance on the test set improved to 78% correct. The new decision boundaries and wrongly labelled training points are shown in figure 2. This improvement may seem surprising, until we remember that the assumptions about the distributions are not very good. In this case we have enough data to estimate a separate covariance matrix for each class, and performance with a Mahalanobis distance is close to that of the discrimination-optimised isotropic distributions.

7 Conclusions

We have shown how some "neural networks" can be understood in terms of parametric statistical pattern recognition, and that modifications of the usual network structures and scoring methods can make them more appropriate for delivering posterior probabilities. Training the network can improve on the equivalent standard model-based parameter estimation methods if the assumptions of the model are incorrect. It remains to be seen whether these modifications make a significant difference in practice, and whether more interesting types of parametric pattern recognition method (such as those based on hidden Markov models, as used in speech recognition) can be cast as adaptive networks.

References

[1] D R Cox and H D Millar. *The Theory of stochastic processes.* Methuen, 1965.

[2] T J Seinowski and C R Rosenberg. *NETtalk: A parallel network that learns to read aloud.* Technical Report JHU/EECS-86/01, Johns Hopkins U. EE&CS, 1986.

[3] L Gillick. Probability scores for backpropagation networks. July 1987. Personal communication.

[4] G E Hinton. *Connectionist Learning Procedures.* Technical Report CMU-CS-87-115, Carnegie Mellon University Computer Science Department, June 1987.

[5] E B Baum and F Wilczek. Supervised learning of probability distributions by neural networks. In D Anderson, editor, *Neural Information Processing Systems*, pages 52–61, Am. Inst. of Physics, 1988.

[6] S Solla, E Levin, and M Fleisher. Accelerated learning in layered neural networks. *Complex Systems*, January 1989.

[7] G.E. Hinton, T.J. Sejnowski, and D.H. Ackley. *Boltzmann machines: constraint satisfaction networks that learn.* Technical report CMU–CS–84–119, Carnegie–Mellon University, May 1984.

[8] E Yair and A Gersho. *The Boltzmann Perceptron Network: a soft classifier.* Technical Report CIPR TR 88-11, Center for Information Processing Research, Dept. of E&CE, UCSB, November 1988.

[9] L R Bahl, P F Brown, P V de Souza, and R L Mercer. Maximum mutual information estimation of hidden Markov model parameters. In *Proc. IEEE ICASSP86*, pages 49–52, 1986.

[10] A J Viterbi. *Principles of Digital Communication and Coding.* McGraw-Hill, 1979.

[11] G E Peterson and H L Barney. Control methods used in a study of vowels. *J. Acoust. Soc. Amer.*, 24(2):175–184, March 1952.

[12] W M Huang and R P Lippmann. Neural net and traditional classifiers. In D Anderson, editor, *Neural Information Processing Systems*, pages 387–396, Am. Inst. of Physics, 1988.