Transforming Neural-Net Output Levels to Probability Distributions

John S. Denker and Yann leCun
AT&T Bell Laboratories
Holmdel, NJ 07733

AT&T Bell Labs Technical Memorandum 11359-901120-05

November 20 1990

Abstract: (1) The outputs of a typical multi-output classification network do not satisfy the axioms of probability; probabilities should be positive and sum to one. This problem can be solved by treating the trained network as a *preprocessor* that produces a *feature vector* that can be further processed, for instance by classical statistical estimation techniques.

(2) We find that in cases of interest, neural networks are (and should be) somewhat *under-determined* because the training data is always limited in quality and quantity. We present a method for computing the first two moments of the probability distribution indicating the range of outputs that are consistent with the input and the training data.

It is particularly useful to combine these two ideas: we implement the ideas of section 1 using Parzen windows, where the shape and relative size of each window is computed using the ideas of section 2. This allows us to make contact between important theoretical ideas (e.g. the ensemble formalism) and practical techniques (e.g. back-prop). Our results also shed new light on and generalize the well-known "softmax" scheme.

1 Distribution of Categories in Output Space

In many neural-net applications, the ultimate goal is a set of C numbers that serve as estimates of the probability of C mutually exclusive outcomes. Important examples include character recognition and other pattern classification applications. According to the axioms of probability, these C numbers should be constrained to be positive and sum to one. We show that rather than modifying the network architecture and/or training algorithm to satisfy this constraint directly, it is advantageous to use a network without the probabilistic constraint, followed by a statistical postprocessor. Similar strategies have been discussed before, e.g. (Fogelman, 1990).

The obvious starting point is a network with C output units. We can train the network with targets that obey the probabilistic constraint, e.g. the target for category "0" is $[1, 0, 0, \cdots]$, the target for category "1" is $[0, 1, 0, \cdots]$, etcetera. This would not, alas, guarantee that the *actual* outputs would obey the constraint. Of course, the actual outputs can always be shifted and normalized to meet the requirement; one of the goals of this paper is to understand the best way to perform such a transformation. A more sophisticated idea would be to construct a network that had such a transformation (e.g. softmax (Bridle, 1990; Rumelhart, 1989)) "built in" even during training. We tried this idea and discovered numerous difficulties, including (1) it is hard to assign target probabilities for the training data, and there were convergence problems if we assigned targets near 1 and 0, (2) it is hard to choose classifier parameters (e.g. softmax gain) during training, and (3) we saw little improvement in using softmax during training, as opposed to plugging a softmax postprocessor onto a network trained without probabilistic constraints. Therefore we will focus attention on networks whose outputs span a C-dimensional hyperspace, and we must decide how to interpret an unnormalized output vector such as $[.6, .7, 0, \cdots]$.

Our analysis is applicable to a wide range of problems. The architecture of the network should be chosen to suit the problem in each case. The network should then be trained using standard techniques. The choice of targets is more or less arbitrary, as will become clear below.

The most principled solution is simply to collect statistics on the trained network. Figures 1, 2 and 3 are scatter plots of output from our Optical Character Recognition ("OCR") network that was trained to recognize the digits "0" through "9". In the first figure, the outputs tend to cluster around the target vectors [the points (T^-, T^+) and (T^+, T^-)], and even though there are a few stragglers, decision regions can be found that divide the space into a high-confidence "0" region, a high-confidence "1" region, and a quite small "rejection" region. In the other two figures, it can be seen that the "3 versus 4" separation is somewhat more difficult than the "0 versus 1" separation, and the "3 versus 5" separation is very challenging.

In all cases, the plotted points indicate the output of the network when the input image is taken from a special "calibration" dataset $\mathcal{L}$ that is distinct both from the training set $\mathcal{M}$ (used to train the network) and from the testing set $\mathcal{G}$ (used to evaluate the generalization performance of the final, overall system).

Given enough training data, we could use a standard statistical technique such as Parzen windows (Duda and Hart, 1973) to estimate the probability density in output space. It is then straightforward to take an unknown input, calculate the corresponding output vector O, and calculate the estimated probability that it belongs to each class:

$$P_{cO}(c|O) = \frac{\rho(c|O)}{\sum_{c'} \rho(c'|O)} \quad \text{for each class } c \tag{1}$$

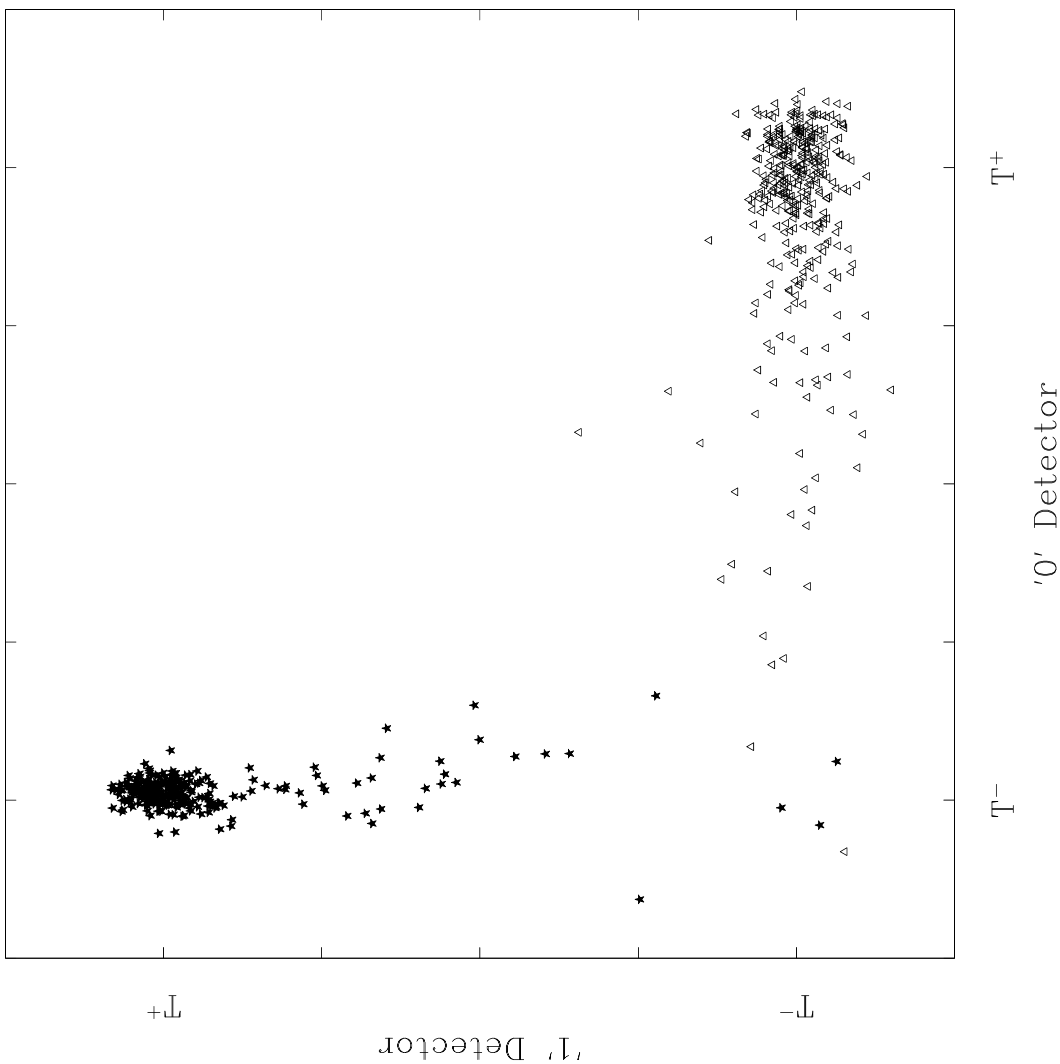

Figure 1: **Scatter Plot: Category 1 versus 0** The open circles represent test images with assigned category c=0, while the + signs represent c=1. The horizontal axis is the activation level of output unit j=0, and the vertical axis is the activation level of output unit j=1; the other 8 axes of output space are suppressed in this projection. The clusters appear elongated because there are so many ways that an item can be neither a “1” or a “0”. This figure contains over 500 points.

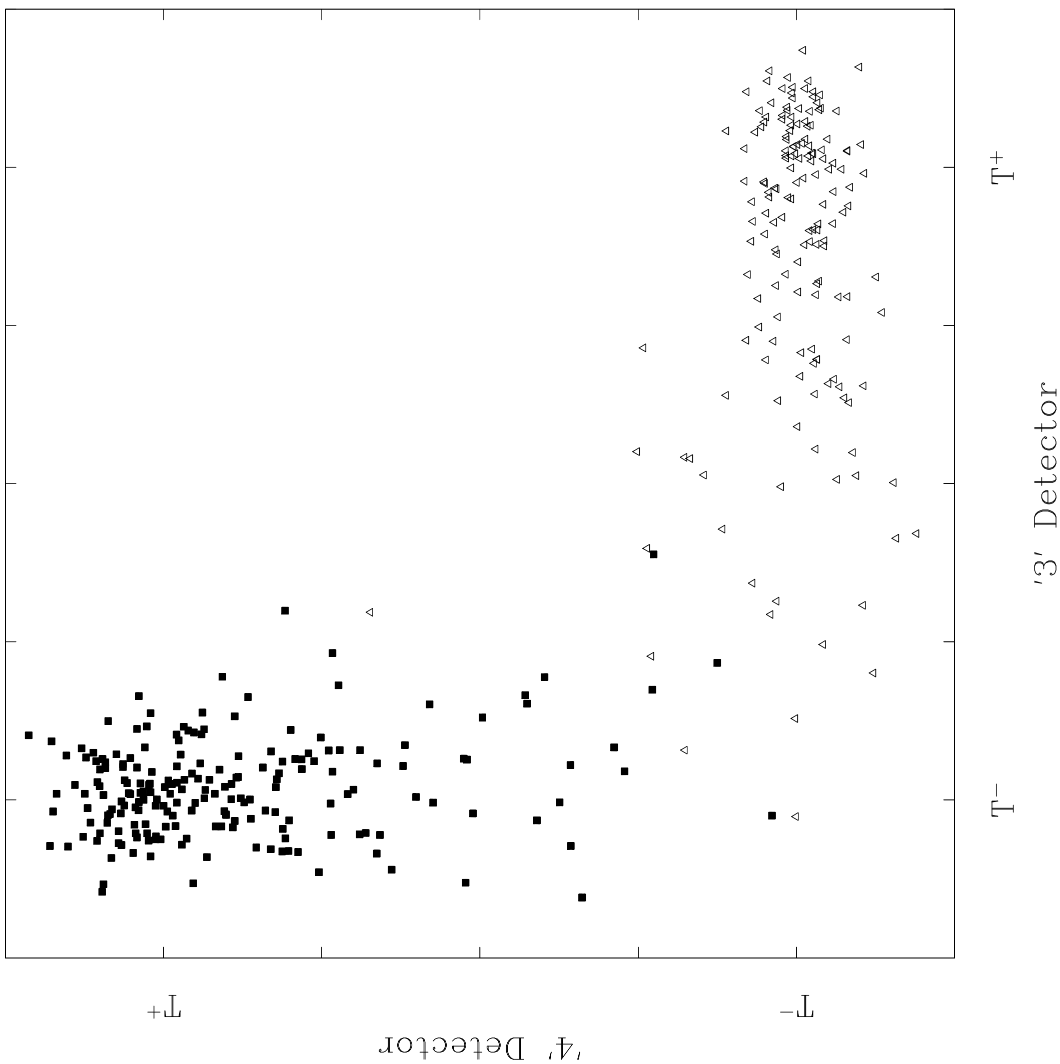

Figure 2: **Scatter Plot: Category 4 versus 3** This is similar to the previous figure. The open triangles represent test images with assigned category c=3, while the filled squares represent c=4.

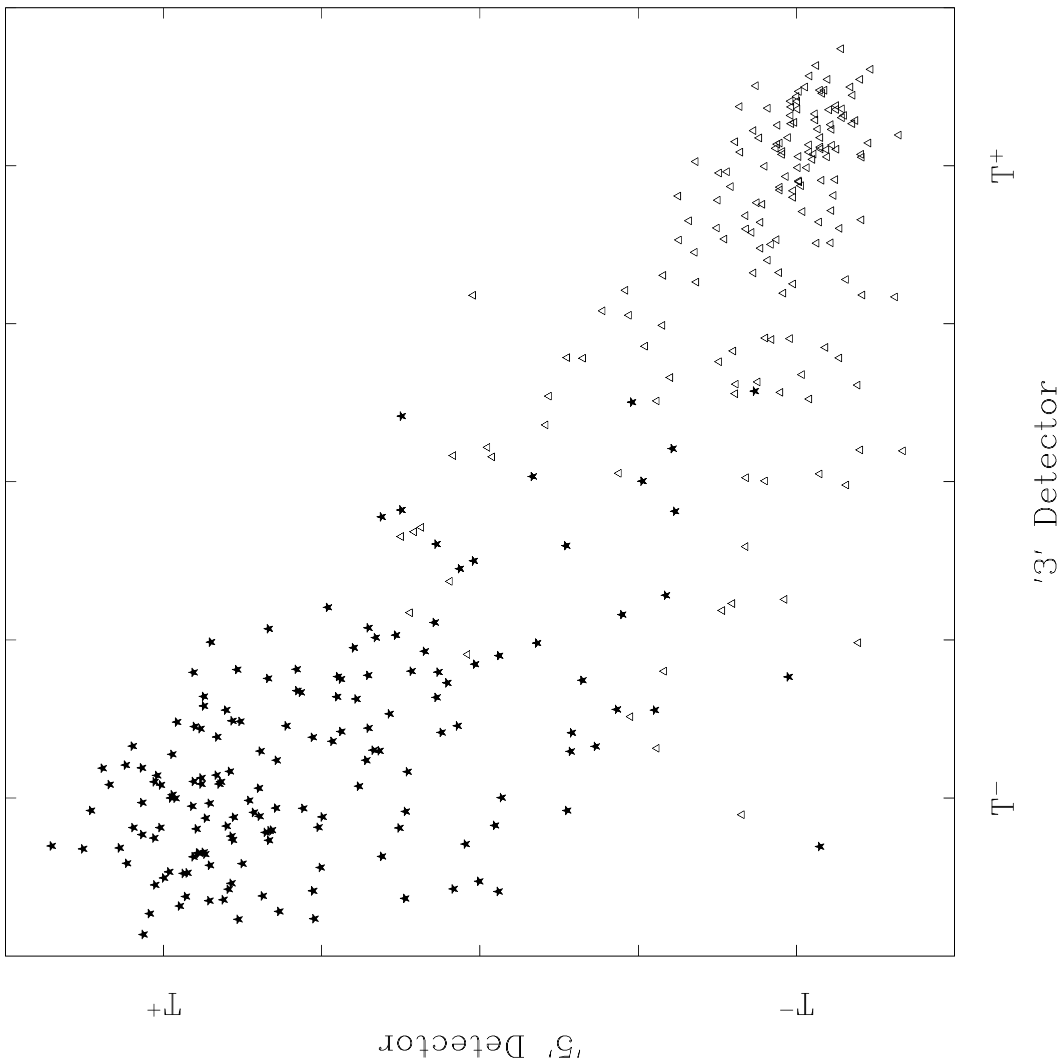

Figure 3: **Scatter Plot: Category 5 versus 3** This is similar to the previous figures. The open triangles again represent category c=3, while the filled stars represent c=5.

where $\rho(c|O)$ is the density (in arbitrary units) of points of category c "at" location O in the scatter plot. The subscript on P_{cO} is to distinguish it from other probabilities (with different functional forms) also denoted by P below.

Technicalities: In principle, it is impossible to measure a probability directly; it is only possible to measure a *frequency* from which the probability can be estimated. In the appropriate limits, we expect frequencies to converge to probabilities. The notion of probability or frequency "at" a point generally requires averaging over a neighborhood of the point; this introduces arcane measurability and smoothness questions which will be discussed below.

Remarks: We note that the dimensionality of the output space need not be exactly equal to the number of classes, but the choice of targets is especially simple in that case. Similarly, since output space is now distinct from probability space, the target values T^+ and T^- need not equal 1 and 0 (and in fact we find it expedient to choose 1 and -1). We also note that methods such as Parzen windows tend to fail when the number of dimensions becomes too large, because it is exponentially harder to estimate probability densities in high-dimensional spaces; this is often referred to as "the curse of dimensionality" (Duda and Hart, 1973). Since the number of output units (typically 10 in our OCR network) is much smaller than the number of input units (typically 400) the method proposed here has a tremendous advantage compared to classical statistical methods applied directly to the input vectors. This advantage is increased by the fact that the distribution of points in network-output space is much more regular than the distribution in the original space.

2 Output Distribution for a Particular Input

The purpose of this section is to calculate how limitations in the quantity and/or quality of training data affect the reliability of neural-net outputs. This calculation does not use the ideas developed in the previous section; the two lines of thought will converge in section 3. The calculation proceeds in two steps: (1) to calculate the range of weight values consistent with the training data, and then (2) to calculate the sensitivity of the output to uncertainty in weight space. The result is a network that not only produces a "best guess" output, but also an "error bar" indicating the confidence interval around that output.

Distribution in Weight Space

A very nice, general formulation of the learning problem is to imagine a probability distribution $P(O, I)$ [rather than the usual function $O = f(I)$] where I and O represent the input vector and output vector respectively. For any specific input pattern, we get a probability distribution $P_{OI}(O|I)$, which can be thought of as a histogram describing the probability of various output values.

Histogram Formalism: Imagine (for starters) a single-output neural network in which the output unit produces a *histogram* rather than a simple number — the output unit could report *all* the possible output values and the corresponding probabilities. Such a network would be ideal for classification problems, especially when the training data was noisy, and/or the inputs were ambiguous. It could be trained by minimizing a loss function E chosen to represent the distance between the desired distribution and the actual distribution reported by the network. This distance can take any of several forms, including L_2 distance, Kullback-Leibler distance, distinguishability distance, etcetera. A particularly interesting version (Milito, 1990) is to require convergence of the *moments* of the histogram to the moments of the desired distribution; different moments could be weighted appropriately and added to form the overall distance measure. Note that vanilla back-prop is a special case of this formalism: training with the usual LMS loss function causes the output unit's activation level to converge to the *mean* (first moment) of the desired probability distribution. This can be thought of as a rather lame sort of "distance between distributions," but it is apparently good enough for many applications.

Point 1: In many applications, such as optical character recognition (LeCun et al., 1990a), there are many elements of the training set for which it is hard to assign non-extremal probabilities. For example, a vertical line (even in the presence of considerable pixel-noise) is assigned to category "1" with probability $1 - \epsilon$, where we have no rational basis for evaluating ϵ. We know ϵ is very small, but setting it to zero leads to considerable theoretical and practical problems.

Point 2: In many cases, the application requires decent estimates of the probabilities of the possible classifications; e.g. if the probability is slightly low, the customer may need to revert to human-based recognition procedures.

Consequence: Observe that point 1, point 2, and the "histogram formalism" (previous paragraphs) are inconsistent. A simple "reproduce the training" scheme cannot get probabilistic outputs from categorical examples. The roots of this inconsistency are quite deep. It is possible to have a case where the training data has absolutely no noise (e.g. when it is generated by a mathematical function on a discrete input space (Denker et al., 1987)). In such a case the output can still be uncertain if the network is underdetermined; the uncertainty arises from lack of data quantity, not quality. In the real world one is faced with *both* problems: less than enough data to (over)determine the network, and less than complete confidence in the data that does exist.

In order to deal with these problems, we need the ensemble viewpoint (Denker et al., 1987). Rather than viewing training as a dynamic process that causes the parameter vector to move around in parameter space, we imagine an ensemble of networks, each with the same architecture but different parameter values. If you wish, imagine one network (one element of the ensemble) located at (and named for) each point W in parameter space. To each of these networks (after looking at m training points) we assign a number $\rho_m(W)$ (Tishby, Levin and Solla, 1989).

We use this number as follows: when it comes time to test the performance of our learning system, we choose a network from the ensemble according to the probability density[1]

$$P_m \equiv P_m(W) := \frac{\rho_m(W)}{\int \rho_m(W')\,\mathrm{d}W'} \tag{2}$$

We average over this probability to get the risk ($\equiv$ expected loss)

$$R = \int E_G(W)\, P_m \mathrm{d}W \tag{3}$$

where $E_G(W)$ is the loss function evaluated on the testing set (the subscript G stands for generalization).

We need to choose a wise value for $\rho_m(W)$ based on the information at hand, namely the training-set loss $E_m(W)$ [an additive function of the m training data], and our *a priori* knowledge of plausible parameter values [contained in $\rho_0(W)$]. We assume all the (training

[1] Notation: we write $A \equiv B$ to indicate that A and B are synonymous; we write $C := D$ to indicate the C is hereby defined to be equal to D.

and testing) data to be independent and identically distributed (iid). Since the loss is additive and probabilities should be multiplicative, we are motivated (Tishby, Levin and Solla, 1989) to choose

$$\rho_m(W) := \rho_0(W) \exp[-\beta E_m(W)] \tag{4}$$

where $\beta \equiv 1/T$ is a measure of our confidence in the training data; T measures our tolerance of error. Note that in the limit $T \to 0$ the exponential is strongly peaked around the minimum-loss (maximum likelihood) point(s) in weight space and $\rho_m(W)$ is strongly peaked at the maximum-a-posteriori (MAP) point, so this formalism contains the more conventional approaches (including our group's earlier efforts (Denker et al., 1987)) as a limiting case. The advantages of a nonzero tolerance T will become clear below; see also (Tishby, Levin and Solla, 1989).

Caveats: The choice of β is important and depends on knowledge, which can be hard to obtain, of the process (i.e. the training/testing data) being modelled. Also, it would be a challenge to show that this procedure is optimal in any sense (or even admissible (Kiefer, 1987)) for any given task — though it is surely no worse than the standard loss-minimization formalisms (since it contains them as a special case). Thirdly, this procedure exists only as a theoretical viewpoint and not as a computational method; since it is infeasible to evaluate the loss function at innumerable points in W space, we rely on methods like gradient descent to find (local) loss-minima when we need them.

In the vicinity of a point (call it $\bar{W}$) that is a local minimum of E, we (LeCun et al., 1990b) can write a Taylor expansion for the loss:

$$E(W) = E(\bar{W}) + \sum_i g_i w_i + \frac{1}{2} \sum_{ij} h_{ij} w_i w_j + O(w^3) \tag{5}$$

where g is the gradient of the loss, h is the second derivative (Hessian), and the relative coordinate in parameter space is $w := W - \bar{W}$. The sums run over the dimensions of parameter space. We now assume that in cases of interest we can neglect the cubic term in this expansion. Also, since E has a (local) minimum at $\bar{W}$, the linear term vanishes. We further assume that we can neglect the off-diagonal terms in the second-order term. We refer to these three assumptions (LeCun et al., 1990b) as the quadratic-extremal-diagonal approximations. Plugging in, we find

$$\rho_m(W) = \rho_0(W) \exp[-\beta \sum_i h_{ii} w_i^2 / 2] \tag{6}$$

We see that the approximations we have introduced correspond to modelling the likelihood (i.e. the data-dependent exponential term on the RHS, with suitable normalization) as a Gaussian centered at the point $\bar{W}$.

Technical aside: we know that the likelihood is not *exactly* a Gaussian everywhere; the hope is that our approximations are valid near $\bar{W}$, and that the likelihood decreases sufficiently quickly that the total likelihood elsewhere is negligible. Actually, the argument is even more complicated, because we know that weight space has certain discrete symmetries associated, for example, with relabelling hidden units and/or flipping the sign of groups of weights (Denker et al., 1987). Therefore there are many points far from $\bar{W}$ where the likelihood becomes large again. Fortunately these points are equivalent to $\bar{W}$ and we can lump them all into an equivalence class; the effect on ρ_m is a constant that drops out when ρ_m is used in equation 2. The general case, with multiple *inequivalent* local maxima of the likelihood, is harder to treat, and is beyond the scope of this paper; for now we make an assumption of non-ergodicity and restrict the analysis to the likelihood peak surrounding $\bar{W}$.

Output Sensitivity

We need to know how the output $O = f_W(I)$ varies as W varies over parameter space (holding the input I fixed). Let us begin by focussing on a the jth component of the output vector. We can always write O_j as

$$O_j(W) = \bar{O}_j + o(W) \tag{7}$$

where $\bar{O} := f_{\bar{W}}(I)$, and the relative coordinate is given (to first order, in the neighborhood of $\bar{W}$) by:

$$o(W) = \sum_k \gamma_k w_k \tag{8}$$

where γ denotes the gradient of o with respect to the parameters W. Note γ, like o, has an implicit dependence on j. It is easy to see that the average $\langle o(W)\rangle_{\rho_m}$ vanishes; this means that to first order, the expected value ("best guess output") of our method is identical to the output of the vanilla network with weight vector $\bar{W}$.

We now calculate the second moment $\sigma := \sqrt{\langle o^2 \rangle}$ as follows:

$$\begin{aligned} \langle o^2 \rangle_{\rho_m} &:= \int o^2(W)\, P_m \mathrm{d}W \\ &= z^{-1} \int (\sum_k \gamma_k w_k)^2 \rho_0(W) \exp[-\beta \sum_i h_{ii} w_i^2/2]\, \mathrm{d}W \end{aligned} \tag{9}$$

where the normalization factor (the denominator in equation 2) is

$$z := \int \rho_0(W) \exp[-\beta \sum_i h_{ii} w_i^2/2]\, \mathrm{d}W \tag{10}$$

If we assume ρ_0 is reasonably constant[2] over the region of interest, this is just a bunch of Gaussian integrals, which are easy to evaluate. The result is rather simple; the uncertainty in the output depends on the curvature of the loss function (h_{ii}), the quality of the training data (β) and the sensitivity of the outputs (γ_i):

$$\sigma^2 = \langle o^2 \rangle_{\rho_m} = \sum_i \frac{\gamma_i^2}{\beta h_{ii}} \tag{11}$$

We now have the first two moments of the output probability distribution ($\bar{O}$ and σ); we could calculate more if we wished.

Note that γ_i depends explicitly on a particular input I, while h_{ii} does not — it depends on a sum over all inputs in the training set. In fact, using the Levenberg-Marquardt approximation to approximate the second derivative by the square of a first derivative, we find

$$h_{ii} \approx 2 \sum_{\substack{\text{training}\\\text{patterns}}} \sum_{\substack{\text{output}\\\text{units}}} \gamma_i^2 \tag{12}$$

which leads to an amusing sum rule for σ^2.

We will denote the conventional Normal (Gaussian) distribution with mean $\bar{x}$ and variance σ by

$$\mathcal{N}[\bar{x}, \sigma](x) := \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{-(x-\bar{x})^2}{2\sigma^2} \tag{13}$$

[2]Treating variations of ρ_0 to first or second order is straightforward.

It is reasonable to expect that the weighted sums (*before* the squashing function) at the last layer of our network are approximately normally distributed, since they are sums of random variables. The signals will surely *not* be normal after passing through a strongly nonlinear squashing function, so we will concentrate on networks where the last-layer units are essentially linear. This can always be arranged by choosing targets that lie in the linear regime of the output units' squashing function. This approximation is easily checked in particular cases. The output distribution is then given by

$$P_{jI}(O_j|I) = \mathcal{N}[\bar{O}, \sigma](O_j) \tag{14}$$

where $\bar{O}$ and σ depend on j and I. This formula applies to a particular output unit j. For multiple output units, we must consider the joint probability distribution $P_{OI}(O|I)$. If the different output units' distributions are independent, P_{OI} can be factored:

$$P_{OI}(O|I) = \prod_j P_{jI}(O_j|I) \tag{15}$$

where the factors are given by equation 14.

We have achieved the goal of this section: we have a formula describing a distribution of outputs consistent with a given input. This is a much fancier statement than the vanilla network's statement that $\bar{O}$ is "the" output. For a network that is not underdetermined, in the limit $T \to 0$, P_{OI} becomes a δ function located at $\bar{O}$, so our formalism contains the vanilla network as a special case. In general, the region where P_{OI} is large constitutes a "confidence region" of size proportional to the fuzziness T of the data and to the degree to which the network is underdetermined.

Note that algorithms exist (Becker and LeCun, 1989), (LeCun et al., 1990b) for calculating γ and h very efficiently — the time scales linearly with the time of calculation of $\bar{O}$. Equation 15 is remarkable in that it makes contact between important theoretical ideas (e.g. the ensemble formalism) and practical techniques (e.g. back-prop).

3 Combining the Distributions

The main objective of this paper is an expression for $P(c|I)$, the probability that input I should be assigned category c. We get it by combining the idea that elements of the calibration set $\mathcal{L}$ are scattered in output space (section 1) with the idea that the network output for each such element is uncertain because the network is underdetermined (section 2). We can then draw a scatter plot in which the calibration data is represented not by zero-size *points* but by *distributions* in output space. One can imagine each element of $\mathcal{L}$ as covering the area spanned by its "error bars" of size σ as given by equation 11. We can then calculate $P(c|I)$ using ideas analogous to Parzen windows, with the advantage that the shape and relative size of each window is calculated, not assumed. This resolves the smoothness and measurability questions raised back in section 1.

To quantify this, we assume the network has already been trained and hold W fixed while we calibrate the statistical postprocessor. We consider the probability $P(O, c, I)$ with which elements are drawn from $\mathcal{L}$. We can always factor this as

$$P(O, c, I) = P_{OI}(O|c, I)P(c, I) \tag{16}$$

where $P_{OI}(O|c, I) \equiv P_{OI}(O|I)$ is given by equation 15 because the output of the network doesn't depend on the category label c assigned to the calibration data.

We assume the probability in input space is large where and only where the training examples are found; this is tantamount to setting the probability (which we can't observe) equal to the frequency (which we can observe):

$$P(c, I) = \frac{1}{L} \sum_{l} \delta(I - I^l)\, \delta^c_{c^l} \tag{17}$$

where $\delta(a - b)$ is the Dirac delta and δ^a_b is the Kronecker delta; I^l and c^l are the input vector and assigned category (respectively) of the lth element of the calibration set $\mathcal{L}$, and L is the cardinality of $\mathcal{L}$. (Once again, we made an assumption here, with no guarantee that it is optimal or even admissible.) This is a very non-smooth distribution in input space; we rely on the smoothness of P_{OI} (the slop in the underdetermined network) to produce a reasonable distribution in output space. Combining equations 16 and 17 and integrating, we find

$$\begin{aligned} P_S(O, c) = \int P(O, c, I)\, \mathrm{d}I &= \frac{1}{L} \sum_{l} \int P_{OI}(O|c, I)\, \delta(I - I^l)\, \delta^c_{c^l}\, \mathrm{d}I \\ &= \frac{1}{L} \sum_{l} P_{OI}(O|c^l, I^l)\, \delta^c_{c^l} \end{aligned} \tag{18}$$

where the subscript S refers to the statistical postprocessor; it assigns categories to points in network-output space according to:

$$P(c|O) = \frac{P_S(O, c)}{P_S(O)} = \frac{P_S(O, c)}{\sum_{c'} P_S(O, c')} = \frac{\sum_{l \in \mathcal{L}^c} P_{OI}(O|c^l, I^l)}{\sum_{l \in \mathcal{L}} P_{OI}(O|c^l, I^l)} \tag{19}$$

where we have introduced $\mathcal{L}^c$ to denote the subset of $\mathcal{L}$ for which the assigned category is c. Hence

$$\begin{aligned} P(c|I) &= \int P(c|O) P_{OI}(O|I)\, \mathrm{d}O \\ &= \int \frac{\sum_{l \in \mathcal{L}^c} P_{OI}(O|I^l)}{\sum_{l \in \mathcal{L}} P_{OI}(O|I^l)}\, P_{OI}(O|I)\, \mathrm{d}O \end{aligned} \tag{20}$$

This is our central result; the rest of the paper will explore its implications. Note that P_{OI} (given by equation 15) is being used in two ways in this formula: to calibrate the statistical postprocessor by summing over the elements of $\mathcal{L}$, and also to calculate the fate of the input I (an element of the testing set).

Our result can be understood by analogy to Parzen windows, although it differs from the standard Parzen windows scheme in two ways. First, it is pleasing that we have a way of calculating the shape and relative size of the windows, namely P_{OI}. Secondly, after we have summed the windows over the calibration set $\mathcal{L}$, the standard scheme would probe $P(c|O)$ at the single point $\bar{O}$ (as in equation 21 below); our expression (equation 20) accounts for the fact that the network's response to the testing input I is blurred over a region given by $P_{OI}(O|I)$ and calls for a convolution.

Correspondence with Softmax

Whenever a new formalism is introduced, it is very illuminating to see under what assumptions it reduces to the previous scheme. We were not surprised that, in suitable limits, our formalism leads to a generalization of the highly useful "softmax" scheme (Bridle, 1990;

Rumelhart, 1989). This provides a deeper understanding of softmax and helps put the present work in context.

The first factor in equation 20, $P(c|O)$, is a perfectly well-defined function of O, but it could be impractical to evaluate it from it definition (summing over the calibration set — equation 19) whenever it is needed. Therefore we sought a closed-form approximation for it.

In the case where the network is not too badly underdetermined, the scatter in figures 1 — 3 will be large compared to the area "covered" by an individual datum. That is, the last factor in the integrand in equation 20 will be so narrowly peaked that it can be treated as a delta function located at $\bar{O}$. (Remember $\bar{O}$ depends implicitly on I via $\bar{O} := f_{\bar{W}}(I)$.) This tempts us to carry out the integration in equation 20, yielding

$$P(c|I) = \frac{\sum_{l\in\mathcal{L}^c} P_{OI}(\bar{O}|I^l)}{\sum_{l\in\mathcal{L}} P_{OI}(\bar{O}|I^l)} \tag{21}$$

We next assume that the density of scatter-plot points in category c can be approximated by multi-dimensional Gaussian, centered on the target vector T^c. In particular let it be a product of Gaussians centered at T_j^c with widths σ_{cj}; as always, c denotes a category, and j denotes a component in output space. In this case equation 21 becomes:

$$P(c|I) = \frac{\prod_j \exp[-(\bar{O}_j - T_j^c)^2/(2\sigma_{cj}^2)]}{\sum_{c'}\prod_j \exp[-(\bar{O}_j - T_j^{c'})^2/(2\sigma_{c'j}^2)]} \tag{22}$$

Writing the targets explicitly as $T_j^c = T^+\delta_j^c + T^-(1-\delta_j^c)$, this can be expanded as

$$\frac{\exp[-\sum_j(\bar{O}_j - T^-)^2/(2\sigma_{cj})]\,\exp[T^\Delta\bar{O}_c/\sigma_{cc}]\,\exp[T^\Delta T^0/\sigma_{cc}]}{\sum_{c'}(\text{same})} \tag{23}$$

where $T^\Delta := T^+ - T^-$, and $T^0 := (T^+ + T^-)/2$. Now we would like to make the additional assumption that the first exponential is independent of c (which would happen if, for each j, σ_{cj} were independent of c). This assumption is *not* supported by the data; a comparison of the horizontal spread of the squares in figure 2 with the horizontal spread of the stars in figure 3 indicates that σ_{43} is much smaller than σ_{53}. If we make the assumption anyway, the first exponential is a common factor in the numerator and denominator, and the whole expression reduces to

$$P(c|I) = \frac{\exp[T^\Delta(O_c - T^0)/\sigma_{cc}^2]}{\sum_{c'}\exp[T^\Delta(O_{c'} - T^0)/\sigma_{c'c'}^2]} \tag{24}$$

This can be compared to the standard softmax expression

$$P(c|I) = \frac{\exp[\Gamma O_c]}{\sum_{c'}\exp[\Gamma O_{c'}]} \tag{25}$$

We see that our formula has three advantages: (1) it is clear how to handle the case where the targets are not symmetric about zero (non-vanishing T^0); (2) the "gain" of the exponentials depends on the category c; and (3) the gains can be calculated from measurable[3] properties of the data. Having the gain depend on the category makes a lot of sense; one can see in the figures that some categories are more tightly clustered than others. One weakness that our equation 24 shares with softmax is the assumption that the output distribution of each output j is circular (i.e. independent of c). This can be remedied by returning to equation 23, if $\mathcal{L}$ is adequate to determine the C^2 different parameters σ_{cj}.

[3]Our formulas contain the overall confidence factor β, which is not as easily measurable as we would like.

Summary

In a wide range of applications, it is extremely important to have good estimates of the probability of correct classification (as well as runner-up probabilities). We have shown how to create a network that computes the parameters a probability distribution (or confidence interval) describing the set of outputs that are consistent with a given input and with the training data. The method has been described in terms of neural nets, but applies equally well to any parametric estimation technique that allows calculation of second derivatives. The analysis outlined here makes clear the assumptions inherent in previous schemes and offers a well-founded way of calculating the required probabilities.

References

Becker, S. and LeCun, Y. (1989). Improving the Convergence of Back-Propagation Learning with Sec ond-Order Methods. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proc. of the 1988 Connectionist Models Summer School*, pages 29–37, San Mateo. Morgan Kaufman.

Bridle, J. S. (1990). Training Stochastic Model Recognition Algorithms as Networks can lead to Maximum Mutual Information Estimation of Parameters. In Touretzky, D., editor, *Advances in Neural Information Processing Systems*, volume 2, (Denver, 1989). Morgan Kaufman.

Denker, J., Schwartz, D., Wittner, B., Solla, S. A., Howard, R., Jackel, L., and Hopfield, J. (1987). Automatic Learning, Rule Extraction and Generalization. *Complex Systems*, 1:877–922.

Duda, R. and Hart, P. (1973). *Pattern Classification And Scene Analysis.* Wiley and Son.

Fogelman, F. (1990). personal communication.

Kiefer, J. (1987). *Introduction to Statistical Inference.* Springer-Verlag, New York.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1990a). Handwritten digit recognition with a back-propagation network. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2 (NIPS*89)*, Denver, CO. Morgan Kaufman.

LeCun, Y., Denker, J. S., Solla, S., Howard, R. E., and Jackel, L. D. (1990b). Optimal Brain Damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2 (NIPS*89)*, Denver, CO. Morgan Kaufman.

Milito, R. (1990). personal communication.

Rumelhart, D. E. (1989). personal communication.

Tishby, N., Levin, E., and Solla, S. A. (1989). Consistent Inference of Probabilities in Layered Networks: Predictions and Generalization. In *Proceedings of the International Joint Conference on Neural Networks*, Washington DC.

It is a pleasure to acknowledge useful conversations with John Bridle.