An abbreviated version of this paper appeared in *Advances in Neural Information Processing Systems 6*, J. Cowan et al. (Ed.'s), Morgan Kauffman, (1994).

Bayesian Backpropagation Over I-O Functions Rather Than Weights

David H. Wolpert
The Santa Fe Institute
1660 Old Pecos Trail
Santa Fe, NM 87501

SFI TR 94-04-019

Abstract

The conventional Bayesian justification for backprop is that it finds the MAP weight vector. As this paper shows, to find the MAP i-o function instead, one must add a correction term to backprop. That term biases one towards i-o functions with small description lengths, and in particular favors (some kinds of) feature-selection, pruning, and weight-sharing. This can be viewed as an *a priori* argument in favor of those techniques.

1 INTRODUCTION

In the conventional Bayesian view of backpropagation (BP) (Buntine and Weigend, 1991; Nowlan and Hinton, 1994; MacKay, 1992; Wolpert, 1993), one starts with the "likelihood" conditional distribution P(training set = t | weight vector **w**) and the "prior" distribution P(**w**). As an example, in regression one might have a "Gaussian likelihood", $P(t \mid \mathbf{w}) \propto \exp[-\chi^2(\mathbf{w}, t)] \equiv \Pi_i \exp[-\{net(\mathbf{w}, t_X(i)) - t_y(i)\}^2 / 2\sigma^2]$ for some constant σ. ($t_X(i)$ and $t_Y(i)$ are the successive input and output values in the training set respectively, and net(**w**, .) is the function, induced by **w**, taking input neuron values to output neuron values.) As another example, the "weight decay" (Gaussian) prior is $P(\mathbf{w}) \propto \exp(-\alpha(\mathbf{w}^2))$ for some constant α.

Bayes' theorem tells us that $P(\mathbf{w} \mid t) \propto P(t \mid \mathbf{w})\, P(\mathbf{w})$. Accordingly, the most probable weight given the data - the "maximum a posteriori" (MAP) **w** - is the mode over **w** of $P(t \mid \mathbf{w})\, P(\mathbf{w})$, which equals the mode over **w** of the "cost function" $M(\mathbf{w}, t) \equiv \ln[P(t \mid \mathbf{w})] + \ln[P(\mathbf{w})]$. So for example with the Gaussian likelihood and weight decay prior, the most probable **w** given the data is the **w** minimizing $\chi^2(\mathbf{w}, t) + \alpha\mathbf{w}^2$. Accordingly BP with weight decay can be viewed as a scheme for trying to find the function from input neuron values to output neuron values (i-o function) induced by the MAP **w**.

One peculiar aspect of this justification of weight-decay BP is the fact that rather than the i-o function induced by the most probable *weight vector*, in practice one would usually prefer to know the most probable i-o *function*. (In few situations would one care more about a weight vector than about what that weight vector parameterizes.) Unfortunately, the difference between these two i-o functions can be large; in general it is ***not*** true that "the most probable output corresponds to the most probable parameter" (Denker and LeCun, 1991).

This paper shows that to find the MAP i-o function rather than the MAP **w** one adds a "correction term" to conventional BP. That term biases one towards i-o functions with small description lengths, and in particular favors feature-selection, pruning and weight-sharing. In this that term constitutes a theoretical argument for those techniques. (A decision-theoretic discussion of the merits of MAP estimators in general can be found in appendix (2).)

Although cast in terms of neural nets, this paper's analysis applies to any case where convention is to use the MAP value of a parameter encoding Q to estimate the value of Q.

2 BACKPROP OVER I-O FUNCTIONS

Assume the net's architecture is fixed, and that weight vectors **w** live in a Euclidean vector space **W** of dimension $|\mathbf{W}|$. Let X be the set of vectors x which can be loaded on the input neurons, and O the set of vectors o which can be read off the output neurons. Assume that the number of elements in X ($|X|$) is finite. This is always the case in the real world, where measuring devices have finite precision, and where the computers used to emulate neural nets are finite state machines. (In addition, in practice we can restrict X to be the input values of the test set, in which case X is explicitly finite. See appendix (3).) For similar reasons O is also finite in practice. However for now assume that O is very large and "fine-grained", and approximate it as a Euclidean vector space of dimension $|O|$. (This assumption usually holds with neural nets, where output values are treated as real-valued vectors.) This assumption will be relaxed later.

Indicate the set of functions taking X to O by Φ. (net(**w**, .) is an element of Φ.) Any $\phi \in \Phi$ is an ($|X|\,|O|$)-dimensional Euclidean vector. Accordingly, densities over **W** are related to densities over Φ by the usual rules for transforming densities between $|\mathbf{W}|$-dimensional and ($|X|\,|O|$)-dimensional Euclidean vector spaces. There are three cases to consider:

1) $|\mathbf{W}| < |X|\,|O|$. In general, as one varies over all **w**'s the corresponding i-o functions net(**w**, .) map out a sub-manifold of Φ having lower dimension than Φ.
2) $|\mathbf{W}| > |X|\,|O|$. There are an infinite number of **w**'s corresponding to each ϕ.
3) $|\mathbf{W}| = |X|\,|O|$. This is the easiest case to analyze in detail. Accordingly I will deal with it first, deferring discussion of cases (1) and (2) until later.

With some abuse of notation, let capital letters indicate random variables and lower case letters indicate values of random variables. So for example **w** is a value of the weight vector random variable **W**. Use 'p' to indicate probability densities. So for example $p_{\Phi|T}(\phi \mid t)$ is the density of the i-o function random variable Φ, conditioned on the training set random variable T, and evaluated at the values $\Phi = \phi$ and T = t.

In general, any i-o function not expressible as net(**w**, .) for some **w** has zero probability. For the other i-o functions, with $\delta(.)$ being the multivariable Dirac delta function,

$$p_{\Phi|T}(\mathrm{net}(\mathbf{w}, .)) = \int d\mathbf{w}'\, p_{\mathbf{W}|T}(\mathbf{w}')\, \delta(\mathrm{net}(\mathbf{w}', .) - \mathrm{net}(\mathbf{w}, .)). \tag{1}$$

When the mapping $\Phi = \mathrm{net}(\mathbf{W}, .)$ is one-to-one, we can evaluate equation (1) to get

$$p_{\Phi|T}(\mathrm{net}(\mathbf{w}, .) \mid t) = p_{\mathbf{W}|T}(\mathbf{w} \mid t) / J_{\Phi,\mathbf{W}}(\mathbf{w}), \tag{2}$$

where $J_{\Phi,\mathbf{W}}(\mathbf{w})$ is the Jacobian of the $\mathbf{W} \to \Phi$ mapping:

$$J_{\Phi,\mathbf{W}}(\mathbf{w}) \equiv \left| \det[\, \partial\Phi_i / \partial\mathbf{W}_j\,](\mathbf{w}) \right| = \left| \det[\, \partial\, \mathrm{net}(\mathbf{w}, .)_i / \partial\mathbf{w}_j\,] \right|. \tag{3}$$

"$\mathrm{net}(\mathbf{w}, .)_i$" means the i'th component of the i-o function net($\mathbf{w}$, .). "net($\mathbf{w}$, x)" means the vector o mapped by net($\mathbf{w}$, .) from the input x, and "$\mathrm{net}(\mathbf{w}, x)_k$" is the k'th component of o. So the "i" in "$\mathrm{net}(\mathbf{w}, .)_i$" refers to a pair of values {x, k}. Each matrix value $\partial\phi_i / \partial\mathbf{w}_j$ is the partial derivative of $\mathrm{net}(\mathbf{w}, x)_k$ with respect to some weight, for some x and k. $J_{\Phi,\mathbf{W}}(\mathbf{w})$ can be rewritten as $\det^{1/2}[g_{ij}(\mathbf{w})]$, where $g_{ij}(\mathbf{w}) \equiv \Sigma_k [\, (\partial\phi_k / \partial\mathbf{w}_i)\, (\partial\phi_k / \partial\mathbf{w}_j)\,]$ is the metric of the $\mathbf{W} \to \Phi$ mapping. This form of $J_{\Phi,\mathbf{W}}(\mathbf{w})$ is usually more laborious to evaluate though.

Unfortunately $\phi = \mathrm{net}(\mathbf{w}, .)$ is not one-to-one; where $J_{\Phi,\mathbf{W}}(\mathbf{w}) \neq 0$ the mapping is *locally* one-to-one, but there are global symmetries which ensure that more than one $\mathbf{w}$ corresponds to each ϕ. Such symmetries arise from permuting the hidden neurons or changing the sign of all weights leading into or out of a hidden neuron - see (Fefferman, 1993) and references therein. (For simplicity, we restrict attention to the usual case (for when $|\mathbf{W}| \leq |X|\,|O|$) where the number of $\mathbf{w}$ corresponding to a particular ϕ is finite.) The easiest way to circumvent the difficulty of this non-injectivity is to make a pair of assumptions.

To begin, restrict attention to $\mathbf{W}_{inj}$, those values $\mathbf{w}$ of the variable $\mathbf{W}$ for which the Jacobian is non-zero. This ensures local injectivity of the map between $\mathbf{W}$ and Φ. Given a particular $\mathbf{w} \in \mathbf{W}_{inj}$, let k be the number of $\mathbf{w}' \in \mathbf{W}_{inj}$ such that net($\mathbf{w}$, .) = net($\mathbf{w}'$, .). (Since net($\mathbf{w}$, .) = net($\mathbf{w}$, .), $k \geq 1$.) Such a set of k vectors form an equivalence class, {$\mathbf{w}$}.

The first assumption is that for all $\mathbf{w} \in \mathbf{W}_{inj}$ the size of {$\mathbf{w}$} (i.e., k) is the same. This will be the case if we exclude degenerate $\mathbf{w}$ (e.g., $\mathbf{w}$'s with all first layer weights set to 0). The second assumption is that for all $\mathbf{w}'$ and $\mathbf{w}$ in the same equivalence class, $p_{\mathbf{W}|D}(\mathbf{w} \mid d) = p_{\mathbf{W}|D}(\mathbf{w}' \mid d)$. This assumption also usually holds. (For example, start with $\mathbf{w}'$ and relabel hidden neurons to get a new $\mathbf{w} \in \{\mathbf{w}'\}$. If we have the Gaussian likelihood and prior, then since neither differs for the two $\mathbf{w}$'s the weight-posterior is also the same for the two $\mathbf{w}$'s.)

Given these assumptions, $p_{\Phi|T}(\mathrm{net}(\mathbf{w}, .) \mid t) = k\, p_{\mathbf{W}|T}(\mathbf{w} \mid t) / J_{\Phi,\mathbf{W}}(\mathbf{w})$. So rather than minimize the usual cost function, M($\mathbf{w}$, t), to find the MAP Φ BP should minimize $M'(\mathbf{w}, t) \equiv M(\mathbf{w}, t) + \ln[\, J_{\Phi,\mathbf{W}}(\mathbf{w})\,]$. The $\ln[\, J_{\Phi,\mathbf{W}}(\mathbf{w})\,]$ term constitutes a correction term to conventional BP. (Note that when the pair of assumptions don't hold, the only change in the analysis is the addition of some book-keeping - for each $\mathbf{w}'$ the correction term now must take into account k and how $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$ varies amongst the elements of {$\mathbf{w}'$}.)

One should not confuse the correction term with other quantities in the neural net literature which involve partial derivative matrices. As an example, one way to characterize the "quality" of a local peak $\mathbf{w}'$ of a cost function involves the Hessian of that cost function (Buntine and Weigend, 1991). The correction term doesn't directly concern the validity of such a Hessian-based quality measure. However it does concern the validity of some

---

[1] As an aside on the subject of Hessians, it should be noted that some workers incorrectly use them

implementations of such a measure. In particular, the correction term changes the location of the peak **w**'. It also suggests that a peak's quality be measured by the Hessian of M'(**w**', t) with respect to ϕ, rather than by the Hessian of M(**w**', t) with respect to **w**.[1] (Note though that calculating the Hessian of M'(**w**', t) with respect to ϕ is usually more difficult than calculating the Hessian of M(**w**', t) with respect to **w** - see appendix (4).)

If we stipulate that the $p_{\Phi|T}(\phi \mid t)$ one encounters in the real world is independent of how one chooses to parameterize Φ, then the probability density of our parameter must depend on how it gets mapped to Φ. This is the basis of the correction term. As this suggests, the correction term won't arise if we use non-$p_{\Phi|T}(\phi \mid t)$-based estimators, like maximum-likelihood estimators. (This is a basic difference between such estimators and MAP estimators with a uniform prior.) The correction term is also irrelevant if it we use an MAP estimate but $J_{\Phi,\mathbf{W}}(\mathbf{w})$ is independent of **w** (as when net (**w**, .) depends linearly on **w**). And even for non-linear net(**w**, .), the correction term has no effect for some non-MAP-based ways to apply Bayesianism to neural nets, like guessing the posterior average Φ (Neal, 1993):

$$E(\Phi \mid t) \equiv \int d\phi \; p_{\Phi|T}(\phi \mid t)\, \phi = \int d\mathbf{w} \; p_{\mathbf{W}|T}(\mathbf{w} \mid t)\, \mathrm{net}(\mathbf{w}, .), \qquad (4)$$

so one can calculate $E(\Phi \mid t)$ by working in **W**, without any concern for a correction term. (Loosely speaking, the Jacobian associated with changing integration variables cancels the Jacobian associated with changing the argument of the probability density. A formal derivation - applicable even when $|\mathbf{W}| \neq |X|\,|O|$ - is in appendix (1).)

One might think that since it's independent of t, the correction term can be absorbed into $p_{\mathbf{W}}(\mathbf{w})$. Ironically, it is precisely because quantities like $E(\Phi \mid t)$ aren't affected by the correction term that this is impossible: Absorb the correction term into the prior, giving a new prior $p^*_{\mathbf{W}}(\mathbf{w}) \equiv d\, p_{\mathbf{W}}(\mathbf{w})\, J_{\Phi,\mathbf{W}}(\mathbf{w})$ (asterisks refers to new densities, and d is a normaliza-

---

in evaluating quantities like output-variances. To understand this, let z denote an element of O, and let $\Phi(x')$ denote the $|O|$ components of Φ associated with some particular input x'. (Recall that Φ is an $(|X| \times |O|)$-dimensional random variable.) Consider $\int dz\, [z - \mathrm{net}(\mathbf{w}', x')]^2\, p_{\Phi(x')|T}(z \mid t)$. This is the variance in the output at x', about the value net(**w**', x'). It gives an idea of how justified you are in estimating the output at x' as net(**w**', x'). The workers in question often wish to evaluate this variance when **w**' is a local maximum of $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$. One way to (try to) do this starts by rewriting the variance as $\int d\phi\, [\phi(x') - \mathrm{net}(\mathbf{w}', x')]^2\, p_{\Phi|T}(\phi \mid t)$, which equals $\int d\mathbf{w}\, [\mathrm{net}(\mathbf{w}, x') - \mathrm{net}(\mathbf{w}', x')]^2\, p_{\mathbf{W}|T}(\mathbf{w} \mid t)$ (whether or not $|\mathbf{W}| = |X|\,|O|$ - see appendix (1)). Up to this point everything's exact. Now approximate the $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$ in the integrand as a Gaussian about the local maximum **w**', expand $[\mathrm{net}(\mathbf{w}, x) - \mathrm{net}(\mathbf{w}', x)]^2$ to low order, and evaluate the resultant Gaussian integral. The resultant value is quoted by some workers as equalling the desired variance. (The Hessian of M(**w**, t) enters the picture in forming the Gaussian approximation to $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$.) The problem with this single-Hessian approximation is that $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$ usually has many local maxima. And the **w**-integral in question is almost always dominated by $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$ *at the other local maxima besides w'*. (This is because the quantity $[\mathrm{net}(\mathbf{w}, x) - \mathrm{net}(\mathbf{w}', x)]^2$ is usually much larger for **w** near those other maxima than it is for **w** near **w**'.) In such a situation, the estimate of the output-variance produced by the single-Hessian approximation is essentially useless. At a minimum, one must examine a representative subset of all the maxima to get a decent estimate of the output variance. (See (Buntine and Weigend, 1991) for a cursory discussion of how to take those other maxima into account.) Unfortunately, when $|\mathbf{W}|$ is large even this doesn't suffice; for such **W** the integral is unlikely to be dominated by the (relatively tiny) part of the space corresponding to the maxima, and one learns nothing by examining those maxima.

tion constant). Then $p^*_{\Phi|T}(\text{net}(\mathbf{w}, .) \mid t) = p_{\mathbf{W|T}}(\mathbf{w} \mid t)$. So by redefining what we call the prior we can justify use of conventional uncorrected BP; the (new) MAP ϕ corresponds to the $\mathbf{w}$ minimizing $M(\mathbf{w}, t)$. However such a redefinition changes $E(\Phi \mid t)$ (amongst other things): $\int d\phi \; p^*_{\Phi|T}(\phi \mid t)\, \phi = \int d\mathbf{w} \; p^*_{\mathbf{W}|T}(\mathbf{w} \mid t) \; \text{net}(\mathbf{w}, .) \neq \int d\mathbf{w} \; p_{\mathbf{W|T}}(\mathbf{w} \mid t) \; \text{net}(\mathbf{w}, .) = \int d\phi \; p_{\Phi|T}(\phi \mid t)\, \phi$. So one can either modify BP (by adding in the correction term) and leave $E(\Phi \mid t)$ alone, or leave BP alone but change $E(\Phi \mid t)$; one can not leave both unchanged.

Moreover, some procedures involve both prior-based modes and prior-based integrals, and therefore are affected by the correction term no matter how $p_{\mathbf{W}}(\mathbf{w})$ is redefined. For example, in the evidence procedure (Wolpert and Strauss, 1994; MacKay, 1992) one fixes the value of a hyperparameter Γ (e.g., α from the introduction) to the value γ' maximizing $p_{\Gamma|T}(\gamma' \mid t)$. Next one finds the value s' maximizing $p_{S|T,\Gamma}(s' \mid t, \gamma')$ for some variable S. Finally, one guesses the ϕ associated with s'. Now it's hard to see why one should use this procedure with $S = \mathbf{W}$ (as is conventional) rather than with $S = \Phi$. But with $S = \Phi$ rather than $\mathbf{W}$, one must factor in the correction term when calculating $p_{S|T,\Gamma}(s \mid t, \gamma')$, and therefore the guessed ϕ is different from when $S = \mathbf{W}$. If one tries to avoid this change in the guessed ϕ by absorbing the correction term into the prior $p_{\mathbf{W}|\Gamma}(\mathbf{w} \mid \gamma)$, then $p_{\Gamma|T}(\gamma \mid t)$ - which is given by an integral involving that prior - changes. This in turn changes γ', and therefore the guessed ϕ again is different. So presuming one is more directly interested in Φ rather than $\mathbf{W}$, one can't avoid having the correction term affect the evidence procedure.

It should be noted that calculating the correction term can be laborious in large nets. One should bear in mind the determinant-evaluation tricks mentioned in (Buntine and Weigend, 1991), as well as others like the identity $\ln[\, J_{\Phi,\mathbf{W}}(\mathbf{w})\,] = \text{Tr}(\ln[\, \partial\phi_i / \partial\mathbf{w}_j\,]) \cong \text{Tr}(\ln^*[\, \partial\phi_i / \partial\mathbf{w}_j\,])$, where $\ln^*(.)$ is $\ln(.)$ evaluated to several orders.

3 EFFECTS OF THE CORRECTION TERM

To illustrate the effects of the correction term, consider a perceptron with a single output neuron, N input neurons and a unary input space: $o = \tanh(\mathbf{w} \cdot \mathbf{x})$, and $\mathbf{x}$ always consist of a single one and N - 1 zeroes. For this scenario $\partial\phi_i / \partial\mathbf{w}_j$ is an $N \times N$ diagonal matrix, and $\ln[\, J_{\Phi,\mathbf{W}}(\mathbf{w})\,] = -2\, \Sigma_{k=1}^{N} \ln[\, \cosh(\mathbf{w}_k)\,]$. Assume the Gaussian prior and likelihood of the introduction, and for simplicity take $2\sigma^2 = 1$. Both $M(\mathbf{w}, t)$ and $M'(\mathbf{w}, t)$ are sums of terms each of which only concerns one weight and the corresponding input neuron. Accordingly, it suffices to consider just the i'th weight and the corresponding input neuron.

Let $\mathbf{x}(i)$ be the input vector which has its 1 in neuron i. Let $o_j(i)$ be the output of the j'th of the pairs in the training set with input $\mathbf{x}(i)$, and m_i the number of such pairs. With $\alpha = 0$ (no weight decay), $M(\mathbf{w}, t) = \chi^2(t, \mathbf{w})$, which is minimized by $\mathbf{w}'_i = \tanh^{-1}[\, \Sigma_{j=1}^{m_i} o_j(i) / m_i]$. If we instead try to minimize $\chi^2(t, \mathbf{w}) + J_{\Phi,\mathbf{W}}(\mathbf{w})$ though, then for low enough m_i (e.g., $m_i = 1$), we find that there is no minimum.[2] The correction term pushes $\mathbf{w}$ away from $\mathbf{0}$, and for low enough m_i the likelihood isn't strong enough to counteract this push.

When weight-decay is used though, modified BP finds a solution, just like unmodified BP

---

[2] Minimizing $\chi^2(t, \mathbf{w}) + J_{\Phi,\mathbf{W}}(\mathbf{w})$ corresponds to finding the MAP ϕ for flat $p_{\mathbf{W}}(\mathbf{w})$. Note that neither the MAP ϕ for flat $p_{\Phi}(\phi)$ nor the MAP $\mathbf{w}$ for flat $p_{\mathbf{W}}(\mathbf{w})$ have the no-minimum problem; both MAP's give the maximum likelihood (ML) ϕ.

does. Since the correction term "pushes out" $\mathbf{w}$, and since tanh(.) grows with its argument, a ϕ found by modified BP has larger (in magnitude) values of o than does the corresponding ϕ found by unmodified BP. In addition, unlike unmodified BP, modified BP has multiple extrema over certain regimes. All of this is illustrated in figures (1) through (3), which graph the value of o resulting from using modified BP with a particular training set t and input value x vs. the value of o resulting from using unmodified BP with t and x. Figure (4) depicts the $\mathbf{w}_i$-dependences of the weight decay term and of the weight-decay term plus the correction term. (When there's no data, BP searches for minima of those curves.)

Now consider multi-layer nets, possibly with non-unary X. Denote a vector of the components of $\mathbf{w}$ which lead from the input layer into hidden neuron K by $\mathbf{w}_{[K]}$. Let $\mathbf{x}'$ be the input vector consisting of all 0's. Then $\partial \tanh(\mathbf{w}_{[K]} \cdot \mathbf{x}') / \partial \mathbf{w}_j = 0$ for any j, $\mathbf{w}$, and K, and for any $\mathbf{w}$, there is a row of $\partial\phi_i / \partial \mathbf{w}_j$ which is all zeroes. This in turn means that $J_{\Phi,\mathbf{W}}(\mathbf{w}) = 0$ for any $\mathbf{w}$, which means that $\mathbf{W}_{inj}$ is empty, and $p_{\Phi|T}(\phi \mid t)$ is independent of the data t. (Intuitively, this problem arises since the o corresponding to $\mathbf{x}'$ can't vary with $\mathbf{w}$, and therefore the dimension of Φ is less than $|\mathbf{W}|$.) So we must forbid such an all-zeroes $\mathbf{x}'$. The easiest way to do this is to require that one input neuron always be on, i.e., introduce a bias unit. An alternative is to redefine Φ to be the functions from the set $\{X - (0, 0, ..., 0)\}$ to O rather than from the set X to O. Another alternative, appropriate when the original X is the set of all input neuron vectors consisting of 0's and 1's, is to instead have input neuron values $\in \{z \neq 0, 1\}$. This is the solution implicitly assumed from now on.

As an aside, one should note that in general this z can not equal -1; due to the symmetries of the tanh, for many architectures z = -1 means that two rows of $\partial\phi_i / \partial \mathbf{w}_j$ are identical up to an overall sign, which means that $J_{\Phi,\mathbf{W}}(\mathbf{w}) = 0$. Intuitively, for those architectures, specifying the output associated with input x leaves us no freedom in what the output associated with input -x is. So in effect, the dimension of Φ is less than that of $\mathbf{W}$.

$J_{\Phi,\mathbf{W}}(\mathbf{w})$ will be small - and therefore $p_\Phi(\text{net}(\mathbf{w}, .))$ will be large - whenever one can make large changes to $\mathbf{w}$ without affecting $\phi = \text{net}(\mathbf{w}, .)$ much. In other words, $p_\Phi(\text{net}(\mathbf{w}, .))$ will be large whenever we don't need to specify $\mathbf{w}$ very accurately. So the correction factor favors those $\mathbf{w}$ which can be expressed with few bits. In other words, the correction factor enforces a sort of automatic MDL (Rissanen, 1986; Nowlan and Hinton, 1994).

More generally, for any multi-layer architecture there are many "singular weights" $\mathbf{w}_{sin} \notin \mathbf{W}_{inj}$ such that $J_{\Phi,\mathbf{W}}(\mathbf{w}_{sin})$ is not just small but equals zero exactly. $p_\mathbf{W}(\mathbf{w})$ must compensate for these singularities, or the peaks of $p_{\Phi|T}(\phi \mid t)$ won't depend on t. So we need to have $p_\mathbf{W}(\mathbf{w}) \to 0$ as $\mathbf{w} \to \mathbf{w}_{sin}$. Sometimes this happens automatically. For example often $\mathbf{w}_{sin}$ includes infinite-valued $\mathbf{w}$'s, since $\tanh'(\infty) = 0$. Because $p_\mathbf{W}(\infty) = 0$ for the weight-decay prior, that prior compensates for the infinite-$\mathbf{w}$ singularities in the correction term.

For other $\mathbf{w}_{sin}$ there is no such automatic compensation, and we have to explicitly modify $p_\mathbf{W}(\mathbf{w})$ to avoid singularities. In doing so though it seems reasonable to maintain a "bias" towards the $\mathbf{w}_{sin}$, that $p_\mathbf{W}(\mathbf{w})$ goes to zero slowly enough so that the values $p_\Phi(\text{net}(\mathbf{w}, .))$ are "enhanced" for $\mathbf{w}$ near $\mathbf{w}_{sin}$. Although a full characterization of such enhanced $\mathbf{w}$ is not in hand, it's easy to see that they include certain kinds of pruned nets (Hassibi and Stork, 1992), weight-shared nets (Nowlan and Hinton, 1994), and feature-selected nets.

To see that (some kinds of) pruned nets have singular weights, let $\mathbf{w}^*$ be a weight vector with a zero-valued weight coming out of hidden neuron K. By (1) $p_\Phi(\text{net}(\mathbf{w}^*, .)) = \int d\mathbf{w}'\, p_\mathbf{W}(\mathbf{w}')\, \delta(\text{net}(\mathbf{w}', .) - \text{net}(\mathbf{w}^*, .))$. Since we can vary the value of each weight $\mathbf{w}^*_i$ leading into neuron K without affecting net($\mathbf{w}$*, .), the integral diverges for nowhere-zero $p_{\mathbf{W}}(\mathbf{w})$. (Recall that δ(net($\mathbf{w}$', .) - net($\mathbf{w}$*, .)) is actually a product of delta functions.) So $\mathbf{w}$* is singular; removing a hidden neuron results in an enhanced probability. This constitutes an *a priori* argument in favor of trying to remove hidden neurons during training.

This argument does not apply to weights leading *into* a hidden neuron; $J_{\Phi,\mathbf{W}}(\mathbf{w})$ treats weights in different layers differently. This fact suggests that however $p_{\mathbf{W}}(\mathbf{w})$ compensates for the singularities in $J_{\Phi,\mathbf{W}}(\mathbf{w})$, weights in different layers should be treated differently by $p_{\mathbf{W}}(\mathbf{w})$. This is in accord with the advice given in (MacKay, 1992).

To see that some kinds of weight-shared nets have singular weights, let $\mathbf{w}$' be a weight vector such that for any two hidden neurons K and K' the weight from input neuron i to K equals the weight from i to K', for all input neurons i. In other words, $\mathbf{w}$ is such that all hidden neurons compute identical functions of x. (For some architectures we'll actually only need a single pair of hidden neurons to be identical.) Usually for such a situation there is a pair of columns of the matrix $\partial\phi_i / \partial\mathbf{w}_j$ which are exactly proportional to one another. (For example, in a 3-2-1 architecture, with $X = \{z, 1\}^3$, $|\mathbf{W}| = |X| \times |O| = 8$, and there are four such pairs of columns.) This means that $J_{\Phi,\mathbf{W}}(\mathbf{w}') = 0$; $\mathbf{w}$' has an enhanced probability, and we have an *a priori* argument in favor of trying to equate hidden neurons during training.

The argument that feature-selected nets have singular weights is architecture-dependent, and there might be reasonable architectures for which it fails. To illustrate the argument, consider the 3-2-1 architecture. Let $x_1(k)$ and $x_2(k)$ with $k = \{1, 2, 3\}$ designate three distinct pairs of input vectors. For each k have $x_1(k)$ and $x_2(k)$ be identical for all input neurons except neuron A, for which they differ. (Note there are four pairs of input vectors with this property, one for each of the four possible patterns over input neurons B and C.) Let $\mathbf{w}$' be a weight vector such that both weights leaving A equal zero. For this situation net($\mathbf{w}$', $x_1(k)$) = net($\mathbf{w}$', $x_2(k)$) for all k. In addition ∂ net($\mathbf{w}$, $x_1(k)$) / $\partial\mathbf{w}_j$ = ∂ net($\mathbf{w}$, $x_2(k)$) / $\partial\mathbf{w}_j$ for all weights $\mathbf{w}_j$ except the two which lead out of A. So k = 1 gives us a pair of rows of the matrix $\partial\phi_i / \partial\mathbf{w}_j$ which are identical in all but two entries (one row for $x_1(k)$ and one for $x_2(k)$). We get another such pair of rows, differing from each other in the exact same two entries, for k = 2, and yet another pair for k = 3. So there is a linear combination of these six rows which is all zeroes. This means that $J_{\Phi,\mathbf{W}}(\mathbf{w}') = 0$. This constitutes an *a priori* argument in favor of trying to remove input neurons during training.

Since it doesn't favor any $p_{\mathbf{W}}(\mathbf{w})$, the analysis of this paper doesn't favor any $p_{\Phi}(\phi)$. Indeed, this paper's analysis implies that setting $p_{\mathbf{W}}(\mathbf{w})$ without any concern for how $\mathbf{w}$ fixes ϕ - which is how many traditional $p_{\mathbf{W}}(\mathbf{w})$ are set - is a dubious way of setting priors.

Nonetheless, when combined with empirical knowledge the analysis of this paper *suggests* certain $p_{\Phi}(\phi)$. For example, there are functions $g(\mathbf{w})$ which empirically are known to be good choices for p_{Φ}(net($\mathbf{w}$, .)) (e.g., $g(\mathbf{w}) \propto \exp[\alpha\mathbf{w}^2]$). There are usually problems with such choices of $p_{\Phi}(\phi)$ though. For example, these $g(\mathbf{w})$ usually make more sense as a prior over $\mathbf{W}$ than as a prior over Φ, which implies that one should instead use p_{Φ}(net($\mathbf{w}$, .)) = $g(\mathbf{w}) / J_{\Phi,\mathbf{W}}(\mathbf{w})$. Moreover it's empirically true that enhanced $\mathbf{w}$ should be favored over other $\mathbf{w}$, as advised by the correction term. So it makes sense to choose a compromise between the two candidate forms for p_{Φ}(net($\mathbf{w}$, .)), $g(\mathbf{w})$ and $g(\mathbf{w}) / J_{\Phi,\mathbf{W}}(\mathbf{w})$. An example of such a compromise is p_{Φ}(net($\mathbf{w}$, .)) $\propto g(\mathbf{w}) / [\, \lambda_1 + \tanh(\lambda_2 \times J_{\Phi,\mathbf{W}}(\mathbf{w}))\,]$ for two hyperparameters $\lambda_1 > 0$ and $\lambda_2 > 0$.

4 BEYOND THE CASE OF BACKPROP WITH $|W| = |X|\,|O|$

When O does not approximate a Euclidean vector space, elements of Φ have probabilities rather than probability densities, and $P(\phi \mid t) = \int d\mathbf{w}\; p_{\mathbf{W}|T}(\mathbf{w} \mid t)\; \delta(\text{net}(\mathbf{w}, .), \phi)$, ($\delta(., .)$ being a Kronecker delta function). Moreover, if O is a Euclidean vector space but $|\mathbf{W}| > |X|\,|O|$, then again one must evaluate a difficult integral; $\Phi = \text{net}(\mathbf{W}, .)$ is not one-to-one so one must use equation (1) rather than (2). In fact, when $|\mathbf{W}| > |X|\,|O|$, even uncorrected BP can be somewhat problematic (e.g., the local minima of unregularized, uncorrected BP generically will be hypersurfaces rather than isolated points, so that the procedure's stopping point isn't well-defined). Fortunately, these two situations are relatively rare.

The final case to consider is $|\mathbf{W}| < |X|\,|O|$ (see section (2)). Let $S(\mathbf{W})$ be the surface in Φ which is the image (under $\text{net}(\mathbf{W}, .)$) of $\mathbf{W}$. For all ϕ $p_\Phi(\phi)$ is either zero (when $\phi \notin S(\mathbf{W})$) or infinite (when $\phi \in S(\mathbf{W})$). So as conventionally defined, "MAP ϕ" is not meaningful.

One way to deal with this case is to embed the net in a larger net, where that larger net's output is relatively insensitive to the values of the newly added weights. An alternative that is applicable when $|\mathbf{W}| / |O|$ is an integer is to reduce X by removing "uninteresting" x's. A third alternative is to consider surface densities over $S(\mathbf{W})$, $p_{S(\mathbf{W})}(\phi)$, instead of volume densities over Φ, $p_\Phi(\phi)$. Such surface densities are given by equation (2), if one uses the metric form of $J_{\Phi,\mathbf{W}}(\mathbf{w})$. (Buntine has emphasized that the Jacobian form is not even defined for $|\mathbf{W}| < |X|\,|O|$, since $\partial\phi_i / \partial\mathbf{w}_j$ is not square then (personal communication).)

As an aside, note that restricting $p_\Phi(\phi)$ to $S(\mathbf{W})$ is an example of the common theoretical assumption that "target functions" come from a pre-chosen "concept class". In practice such an assumption is usually ludicrous - whenever it is made there is an implicit hope that it constitutes a valid approximation to a more reasonable $p_\Phi(\phi)$.

When decision theory is incorporated into Bayesian analysis, only rarely does it advise us to evaluate an MAP quantity (i.e., use BP). Instead Bayesian decision theory usually advises us to evaluate quantities like $E(\Phi \mid t) = \int d\phi\; p_{\Phi|T}(\phi \mid t)\; \phi$ (see appendix (2)). Just as it does for the use of MAP estimators, the analysis of this paper has implications for the use of such $E(\Phi \mid t)$ estimators. In particular, if we don't modify the prior to remove the singularities in the correction term (see previous section), then those singularities occur in the integrand of $E(\Phi \mid t)$. Accordingly, one would expect the ϕ's around those singularities to be "major contributors" to the integral. (Whether or not they dominate the integral depends on the precise mapping from $\mathbf{W}$ to Φ.)

A related implication for Bayesian decision theory comes from the fact that one way to evaluate $E(\Phi \mid t) = \int d\mathbf{w}\; p_{\mathbf{W}|T}(\mathbf{w} \mid t)\; \text{net}(\mathbf{w}, .)$ is to expand $\text{net}(\mathbf{w}, .)$ to low order and then approximate $p_{\mathbf{W}|T}(\mathbf{w} \mid t)$ as a sum of Gaussians (Buntine and Weigend, 1991). For the case where $|\mathbf{W}| = |X|\,|O|$ and we've modified the prior to remove singularities, equation (4) suggests an alternative: write $E(\Phi \mid t)$ as $\int d\phi\; p_{\Phi|T}(\phi \mid t)\; \phi$ and approximate $p_{\Phi|T}(\phi \mid t)$ as a sum of Gaussians, as discussed in appendix (4). (Note that if $|\mathbf{W}| < |X|\,|O|$ then $p_{\Phi|T}(\phi \mid t)$ can't be approximated as a Gaussian.) Since fewer approximations are used (no low order expansion of $\text{net}(\mathbf{w}, .)$), this might be more accurate. Of course, if the dimension of the space over which we're integrating is large, no scheme based on behavior around the peaks of some terms in the integrand is likely to give a good approximation.

Appendix One: The irrelevance of choice of integration variables for calculating moments.

This appendix reviews the standard proof that expectation values are "independent" of the choice of integration variables, i.e., that there's no correction term for calculating moments. The result is recapitulated here due to its importance for this paper's results.

Let "Pr" indicate probabilities, and "p" indicate probability densities (as in the text). Let $\mathbf{U}$ be a $|\mathbf{U}|$-dimensional real-valued random variable, and $\mathbf{V}$ a $|\mathbf{V}|$-dimensional random variable. A particular value of $\mathbf{V}$ is indicated by $\mathbf{v}$, and a particular value of $\mathbf{U}$ is indicated by $\mathbf{u}$. In addition to $\mathbf{U}$ and $\mathbf{V}$, we have a random variable $\mathbf{D}$ (with values $\mathbf{d}$) which will be used for conditioning probability densities. The results claimed in the text obtain if one substitutes $\mathbf{U}$ for $\mathbf{W}$, and $\mathbf{T}$ for $\mathbf{D}$ and either F or Φ (as the case might be) for $\mathbf{V}$.

It is assumed that $\mathbf{V}$ is given in terms of $\mathbf{U}$ via $\mathbf{V}_i = \mathrm{net}_i(\mathbf{U})$ (i is a component of $\mathbf{V}$). No other assumptions are made. In particular, no assumptions are made about the single-valuedness (or even the existence, for all $\mathbf{V}$) of the mapping from $\mathbf{V}$ to $\mathbf{U}$, and no assumption is made about the relative sizes of $|\mathbf{U}|$ and $|\mathbf{V}|$.

We want to evaluate $E(Z(\mathbf{V}) \mid \mathbf{d}) = \int d\mathbf{v}\, Z(\mathbf{v})\, p_{\mathbf{V}|\mathbf{D}}(\mathbf{v} \mid \mathbf{d})$ for some function Z(.). (In most of this paper, $Z(\mathbf{v}) = \mathbf{v}$.)

First write $p_{\mathbf{V}|\mathbf{D}}(\mathbf{v} \mid \mathbf{d}) = \partial_{\mathbf{v}_1} \ldots \partial_{\mathbf{v}_{|\mathbf{V}|}} \{ \Pr(\mathbf{V}_1 < \mathbf{v}_1, \ldots, \mathbf{V}_{|\mathbf{V}|} < \mathbf{v}_{|\mathbf{V}|} \mid \mathbf{d}) \}$.

Then expand $\Pr(\mathbf{V}_1 < \mathbf{v}_1, \ldots, \mathbf{V}_{|\mathbf{V}|} < \mathbf{v}_{|\mathbf{V}|} \mid \mathbf{d}) = \int d\mathbf{u}\, [\, p_{\mathbf{U}|\mathbf{D}}(\mathbf{u} \mid \mathbf{d}) \times \Pi^{|\mathbf{V}|}_{i=1}\ \theta(\mathbf{v}_i - \mathrm{net}_i(\mathbf{u}))\,]$,

where $\theta(.)$ is the Heaviside step function.

Interchange orders of integration:

$$\int d\mathbf{v}\, Z(\mathbf{v})\, p_{\mathbf{V}|\mathbf{D}}(\mathbf{v} \mid \mathbf{d}) = \int d\mathbf{v}\, Z(\mathbf{v})\ \partial_{\mathbf{v}_1} \ldots \partial_{\mathbf{v}_{|\mathbf{V}|}} \{ \int d\mathbf{u}\, [\, p_{\mathbf{U}|\mathbf{D}}(\mathbf{u} \mid \mathbf{d}) \times \Pi^{|\mathbf{V}|}_{i=1}\ \theta(\mathbf{v}_i - \mathrm{net}_i(\mathbf{u}))\,]\, \}$$
$$= \int d\mathbf{u} [\, p_{\mathbf{U}|\mathbf{D}}(\mathbf{u} \mid \mathrm{d}) \times \int d\mathbf{v}\, Z(\mathbf{v})\ \partial_{\mathbf{v}_1} \ldots \partial_{\mathbf{v}_{|\mathbf{V}|}} \{ \Pi^{|\mathbf{V}|}_{i=1}\ \theta(\mathbf{v}_i - \mathrm{net}_i(\mathbf{u}))\, \}\,].$$

Evaluating the partial derivatives, we get

$$E(Z(\mathbf{V}) \mid \mathbf{d}) = \int d\mathbf{u}\, [\, p_{\mathbf{U}|\mathbf{D}}(\mathbf{u} \mid \mathbf{d}) \times \int d\mathbf{v}\, Z(\mathbf{v})\, \Pi^{|\mathbf{V}|}_{i=1}\ \delta(\mathbf{v}_i - \mathrm{net}_i(\mathbf{u}))\,]$$
$$= \int d\mathbf{u}\, [\, p_{\mathbf{U}|\mathbf{D}}(\mathbf{u} \mid \mathbf{d}) \times Z(\mathrm{net}_1(\mathbf{u}), \mathrm{net}_2(\mathbf{u}), \ldots, \mathrm{net}_{|\mathbf{V}|}(\mathbf{u}))\,].$$

This gives our final result:

$$E(Z(\mathbf{V}) \mid \mathbf{d}) = \int d\mathbf{v}\, p_{\mathbf{V}|\mathbf{D}}(\mathbf{v} \mid \mathbf{d})\, Z(\mathbf{v}) = \int d\mathbf{u}\, [\, p_{\mathbf{U}|\mathbf{D}}(\mathbf{u} \mid \mathbf{d}) \times Z(\mathrm{net}_1(\mathbf{u}), \mathrm{net}_2(\mathbf{u}), \ldots, \mathrm{net}_{|\mathbf{V}|}(\mathbf{u}))\,].$$

Appendix Two: Bayesian decision theory and neural nets.

This appendix casts BP in terms of (predictive paradigm) Bayesian decision theory (Berger, 1985; Titterington, 1985), using a variation of the formalism in (Wolpert, 1992). To illustrate the discussion, this appendix will examine both an example where O is the direct object of interest (e.g., when one uses neural nets for regression) and an example where O gives relative probabilities of possible values of the direct object of interest (e.g., when one uses neural nets for classification in concert with a scheme like softmax). The first part of the appendix presents a cursory overview of (the salient aspects of) decision theory, the second section relates decision theory to neural nets, and the third section uses this to discuss various kinds of MAP estimators involving neural nets.

Wherever possible "P" notation will be used: the arguments of a "P" indicates if it's a probability or a density, and if the latter, what random variable it's over.

1) A quick review of Bayesian decision theory.

Assume we have an input space X and an output space Y. For expository simplicity, I will take both to be finite (although sometimes Y will implicitly be assumed to be a very large set of closely spaced real numbers). In general, Y is <u>not</u> the same as the space O discussed in the text - the space O will enter the analysis later.

Let 'f' be a function giving the probability of $y \in Y$ conditioned on $x \in X$. This is indicated by writing $P(y \mid x, f) = f_{x,y}$. (Note that f is a vector of real numbers, with components $f_{x,y}$.) When extra precision is required, 'F' will indicate the random variable of which 'f' is an instantiation. So for example, $p_{Y|X,F}(y \mid x, f) = f_{x,y}$.

f labels the "true" or "target" conditional distribution of y given x, in that the training set is generated according to f. In other words, $P(t \mid f)$ is our likelihood function. For the purposes of this paper, this means that $P(t_Y \mid t_X, f) = \Pi_i f_{t_X(i), t_Y(i)}$. Of course, we don't know f, only the training set t - loosely speaking, our goal is to use t to guess f.

Let $h_{x,y}$ be the x-conditioned probability distribution over values y which is produced by our learning algorithm in response to t. Sometimes our algorithm's "output" is a quantity based on h (e.g., our algoirthm might produce a decision of some sort based on h). But often h itself is the output of our algorithm. In particular, if our algorithm's output is a guessed function from X to Y, and if our algorithm always guesses the same function when trained on the same data t, then $h_{x,y}$ is of the form $\delta(y, \eta(x, t))$ for some function $\eta(., .)$. (See examples (1) and (2) below.)

Now define a real-world (usually random variable) quantity C which represents the "loss" (or what in some circumstances is called "cost" or "utility" or "value") associated with a particular $f_{x,y}$ and $h_{x,y}$. For each t we want our learning algorithm to produce the associated "Bayes-optimal" $h_{x,y}$, which is the $h_{x,y}$ minimizing the posterior expected loss

$$E(C \mid h, t) \equiv \int df \int d(\text{loss values } c)\ [c\ p_{C|H,F,T}(c \mid h, f, t)\ p_{F|H,T}(f \mid h, t)].$$

Now in supervised learning the output of a learning algorithm h and the target f are only coupled through the training set t, i.e., if we know t, knowing h as well doesn't help in nailing down f. Accordingly (see (Wolpert, 1992)), $p_{F|H,T}(f, \mid h, t) = p_{F|T}(f \mid t)$, and

$$E(C \mid h, t) \equiv \int df \int dc\ [c\ p_{C|H,F,T}(c \mid h, f, t)\ p_{F|T}(f \mid t)].$$

In the P notation this is written as

$$E(c \mid h, t) \equiv \int df\ dc\ [c\ P(c \mid h, f, t)\ P(f \mid t)].$$

So long as we can identify a distribution h produced by our learning algorithm and so long as the real-world cost can be (perhaps only statistically) related to f, h, and t, the goal of finding the h minimizing $E(c \mid h, t)$ is well-defined. Note that to meet this goal we need to know $P(c \mid h, f, t)$. This distribution in effect defines what we mean by "loss". It is set by the supervised learning scenario at hand, and by how the end-user uses h.

As an example - one followed for most of the rest of this appendix - often when we're given h and f, to find a value of the random variable C we: i) randomly choose an x value x' according to some pre-fixed distribution $\pi(x)$; ii) randomly sample the y-dependence of $h_{x',y}$ to get a guessed y value, y_h; iii) randomly sample the y dependence of $f_{x',y}$ to get a "true" or "test set" y value y_f; and finally iv) set c to the value $L(y_h, y_f)$ for some function $L(., .)$.[3] $L(., .)$ is the loss function for this procedure, and should reflect the real-world cost of guessing y_h when the "true" y value is y_f. Typical pedagogical choices are the quadratic loss, $L(a, b) = (a - b)^2$, and the zero-one loss, $L(a, b) = 1 - \delta(a, b)$.

With this procedure for generating values of c, $P(c \mid h, f, t)$ is given by the sum $\Sigma_{x,y_h,y_f}\ \pi(x)\ P(y_f \mid x, f)\ P(y_h \mid x, h)\ \delta(c, L(y_h, y_f))$. This means that

$$E(c \mid h, t) = \Sigma_{x,y_h,y_f} \{ \pi(x)\ h_{x,y_h}\ L(y_f, y_h) \int df\ f_{x,y_f}\ P(f \mid t) \}.$$

In the context of this $P(c \mid h, f, t)$, with slight abuse of convention the term "predictive distribution" will mean $\int df\ f_{x,y_f}\ P(f \mid t)$. I will write this integral as $\boldsymbol{P}(y_f, x, t)$. Often x is statistically independent of t, in which case $\boldsymbol{P}(y_f, x, t) = P(y_f \mid x, t)$, so that $\boldsymbol{P}(y_f, x, t)$ "predicts" the probability of the test set output y_f, given training set t and test set input x.

Our equation for $E(c \mid h, t)$ should be familiar - it closely resembles what in the neural net community is sometimes defined as "generalization error", or (average) "test set error". Indeed, for our $P(c \mid f, h, t)$, as it's usually defined the generalization error is identical to $E(c \mid h, f) = E(c \mid f, h, t) = \Sigma_{x,y_h,y_f} \{ \pi(x)\ h_{x,y_h}\ L(y_f, y_h)\ f_{x,y_f} \}$.[4] Accordingly, $E(c \mid h, t)$ is just the (posterior) average, over all f, of the generalization error for a particular h and f. The Bayes-optimal h is simply the h that minimizes this posterior expected generalization

---

[3] Note the assumption in step (iii) that the test set is generated from x values and f in the same way that t is. In general, $P(t_Y(i) \mid t_X(i), f) = f_{t_X(i),\ t_Y(i)}\ \forall\ i$, but P(test set output y' | test set input x', f) $\neq f_{y',x'}$ if the process generating the test set is different from the process generating the training set t.

[4] It's important to understand that this "generalization error" form for $E(c \mid f, h, t)$ is not *a priori* necessary in any sense. Many other forms are reasonable. As an example, in an extension of the scenario considered in (Rosen, 1994), a value of c is generated as follows: The distribution f is sampled to get y_f, and $\pi(x)$ is sampled to get x, as usual. The learner then provides $h_{x,y}$ - generated in response to t - to a distribution of "users". Each user, following his own algorithm, uses $h_{x,y}$ and x to make a "decision" $u \in U$. Each such user's u is combined with y_f to get a "utility" value. The

error.

It's easy to verify that for the $E(c \mid h, t)$ given above, to produce the Bayes-optimal h your algorithm must i) take each x and find $y^*(x, t)$, the y_h value that minimizes the sum $\Sigma_{y_f} L(y_f, y_h)\, \boldsymbol{P}(y_f, x, t)$ for that x; and then ii) output $h_{x,y} = \delta(y, y^*(x, t))$. In other words, for any x optimal behavior is to guess the Y value $y^*(x, t)$. Since the Bayes-optimal h is fixed by the loss function and the predictive distribution, and since the user specifies the loss function, all of the statistical inference reduces to finding the predictive distribution.

As an example, by setting $\partial / \partial y_h\, [\Sigma_{y_f} L(y_f, y_h)\, \boldsymbol{P}(y_f, x, t)] = 0$, we see that for quadratic loss $y^*(x, t) = \Sigma_{y_f} y_f\, \boldsymbol{P}(y_f, x, t)$. (When the predictive distribution equals $P(y_f \mid x, t)$, this is just the posterior expected y, $E(y_f \mid x, t)$.) As another example, for zero-one loss $y^*(x, t) = \text{argmax}_{y_f} \boldsymbol{P}(y_f, x, t)$.

2) Decision theory and neural nets.

To relate the foregoing to neural nets, we need to cast neural net weight vectors **w** as parameters of X-Y relationships f. Accordingly, we must specify the coordinate transformation taking (Euclidean vectors) **w** to (Euclidean vectors) f. In the examples of this presented below, attention is implicitly restricted to neural nets with continuously differentiable activation functions.

Example 1: In schemes like softmax (Bridle, 1990), **w** directly parameterizes a conditional distribution f, with the output neuron index giving y. More formally, in these schemes **w** fixes f via the rule $f_{x,y}(\mathbf{w}) = \text{net}(\mathbf{w}, x)_y$, where by "net(**w**, x)" is meant the vector of the values of output neurons which arises when input x is propagated through a net with weight vector **w**, and the subscript 'y' denotes the y'th output neuron. (N.b. $\text{net}(\mathbf{w}, x)_y$ must be non-negative and normalized over y for this probabilistic interpretation of the net's output to be valid; to ensure this "net(**w**, x)" is usually a neural net used with a bit of post-processing.)

Since there can be f's which don't correspond to any **w** (or perhaps correspond to more than one **w**), one must be careful in converting $\boldsymbol{P}(y, x, t)$ from an f-space integral to a w-space integral. Following along with the standard procedure (see appendix (1)), one gets

$$\boldsymbol{P}(y, x, t) = \int d\mathbf{w}\, \text{net}(\mathbf{w}, x)_y\, P(\mathbf{w} \mid t) = [\, \int d\mathbf{w}\, \text{net}(\mathbf{w}, x)\, P(\mathbf{w} \mid t)\,]_y.$$

So as an example, for this scenario, with zero-one loss, the Bayes-optimal guess is given by $y^*(x, t) = \text{argmax}_y \int d\mathbf{w}\, \text{net}(\mathbf{w}, x)_y\, P(\mathbf{w} \mid t)$.

The usual next step is to set the likelihood $P(t \mid \mathbf{w})$ and prior $P(\mathbf{w})$, and plug these into

---

value of c - the loss - is the average of the utility over the entire set of users. Write that loss as $L^*(h, y_f, x)$. For this scenario $P(c \mid f, h, t) = \Sigma_{x,y_f} \pi(x)\, P(y_f \mid x, f)\, \delta(c, L^*(h, y_f, x))$, which means that $E(c \mid f, h, t) = \Sigma_{x,y_f} \pi(x)\, P(y_f \mid x, f)\, L^*(h, y_f, x)$. For many choices of $L^*(., ., .)$ this $E(c \mid f, h, t)$ can not be written as $\Sigma_{x,y_h,y_f} \pi(x)\, P(y_f \mid x, f)\, P(y_h \mid x, h)\, L(y_f, y_h)$ for some function $L(., .)$. QED.

$P(\mathbf{w} \mid t) \propto P(t \mid \mathbf{w})\, P(\mathbf{w})$ to fix $\boldsymbol{P}(y, x, t)$ and therefore $y^*(x, t)$. As an example, one popular choice is the weight-decay prior, $P(\mathbf{w}) \propto \exp(-\alpha \mathbf{w}^2)$ for some constant α. Another popular choice is to write $P(t \mid \mathbf{w}) = P(t_X \mid \mathbf{w})\, P(t_Y \mid t_X, \mathbf{w})$, and assume $P(t_X \mid \mathbf{w})$ has no **w**-dependence. (The general implications of having $P(t_X \mid \mathbf{w})$ be independent of **w** are discussed in (Wolpert, 1994).) Then one takes $P(t_Y \mid t_X, \mathbf{w}) \propto \Pi_i\, P(t_Y(i) \mid t_X(i), \mathbf{w})$, and $P(t_Y(i) \mid t_X(i), \mathbf{w}) = \mathrm{net}(\mathbf{w}, t_X(i))_{t_y(i)}$.

Having set $P(\mathbf{w} \mid t)$ and fixed the loss function, we've completely specified how we want our learning algorithm to behave; we want it to guess the corresponding Bayes-optimal h.

Example 2: Consider schemes like those which use neural nets for "regression" (i.e., surface-fitting). In these schemes the neural net output is not viewed as a direct conditional probability distribution $f_{x,y}$, but rather as a Y value to which noise is added to get $f_{x,y}$. So **w** doesn't directly code for a $P(y \mid x)$, but rather for a mapping from an X value to a Y value. This means that $\mathrm{net}(\mathbf{w}, x)$ is a Y value, and we have (as an example) the y-dependence of $f_{x,y}(\mathbf{w}) \sim N(\mathrm{net}(\mathbf{w}, x), \sigma^2)$, for some pre-fixed σ.[5]

Intuitively, this is the familiar "function plus noise" scenario; we can view $f_{x,y}(\mathbf{w})$ as the distribution generating training set output values when the "target" input-output function is given by the weight vector **w**. For this scenario,

$$\boldsymbol{P}(y, x, t) = \int d\mathbf{w}\, \exp[-(\mathrm{net}(\mathbf{w}, x) - y)^2 / 2\sigma^2]\, P(\mathbf{w} \mid t),$$

up to overall normalization constants. So as an example, if we have quadratic loss, then $y^*(x, t) = \int d\mathbf{w}\, \mathrm{net}(\mathbf{w}, x)\, P(\mathbf{w} \mid t)$, independent of σ. (Note that we would not get so neat a $y^*(x, t)$ if we used zero-one loss, like in example (1).)

As before, to set $P(\mathbf{w} \mid t)$ we might have $P(\mathbf{w}) \propto \exp(-\alpha \mathbf{w}^2)$. However now we would probably have something like a gaussian noise likelihood, $P(t \mid \mathbf{w}) \propto P(t_Y \mid t_X, \mathbf{w}) \propto \Pi_i \exp[-\{\mathrm{net}(\mathbf{w}, t_X(i)) - t_y(i)\}^2 / 2(\sigma_L)^2] \equiv \exp[-\chi^2(\mathbf{w}, t)]$.[6]

---

[5] So that statements like "the y-dependence of $f_{x,y}(\mathbf{w}) \sim N(\mathrm{net}(\mathbf{w}, x), \sigma^2)$" can be meaningful, in this example it is implicitly assumed that although it is finite, Y is very large and consists of closely spaced element of $\mathbf{R}^N$ for some N. Since we also are assuming that "$\mathrm{net}(\mathbf{w}, x)$ is a Y value", $O \subseteq Y$, and strictly speaking the set of possible output values of the neural net is finite. (For the reasons discussed in section two this finiteness of O presents no foundational difficulties.) For simplicity, I will assume that $O = Y$; since Y approximates $\mathbf{R}^N$, so does O (again see section two). Note that despite all these finite-but-large issues, both **w** and $f_{x,y}$ are still taken to be infinite-precision real-valued numbers. Note also the notational convention that "$|O|$" means N.

[6] Recall that our general form for $P(c \mid h, f, t)$ assumes that training and testing sets are generated identically, which means that σ_L must equal σ. However often when dealing with function+noise scenarios we wish to guess the target function directly (rather than that function with noise added). To deal with this formally, one must change $P(c \mid h, f, t)$ so that P(test set output value y_f | test set input value x, f) does not equal f_{x,y_f}, but rather equals a (y-space) delta function centered at the mode of (the y-dependence of) the Gaussian function $f_{x,y}$. This is equivalent to setting σ to 0, inde-

See (Buntine and Weigend, 1991) for other possible choices of the prior and likelihood. Just as in example (1), once the loss function and $P(\mathbf{w} \mid t)$ are set, we've completely specified the goal of our learning algorithm.

3) Decision theory, neural nets, and MAP estimators.

Summarizing, the y*'s of both our examples are determined by the vector of $|X|$ values $\int d\mathbf{w}\ \mathrm{net}(\mathbf{w}, x)\ P(\mathbf{w} \mid t)$. In the case of example (1), this is true independent of the loss function. In case of example (2), this integral arises only because of the loss function we chose for that example, and because we took $f_{x,y}(\mathbf{w})$ to be symmetric in Y about $\mathrm{net}(\mathbf{w}, x)$.

Unfortunately though, we can't readily evaluate $\int d\mathbf{w}\ \mathrm{net}(\mathbf{w}, x)\ P(\mathbf{w} \mid t)$ and therefore can't readily find our Bayes-optimal guess. (See (Neal, 1993) for advice on how to approximate the integral via Monte Carlo methods.) One crude way out of this difficulty is to hope that the mode of $P(\mathbf{w} \mid t)$ well approximates its mean, and that that MAP $\mathbf{w}$ provides a similarly good approximation for finding y*. So for instance in example (2), these hopes would direct us to find the $\mathbf{w}$, $\mathbf{w}^*$, which minimizes $\chi^2(\mathbf{w}, t) + \alpha\mathbf{w}^2$. We would then interpret $\mathrm{net}(\mathbf{w}^*, x)$ as (an approximation to) the Bayes-optimal guess for the function to which noise was added to generate our training set. This is the implicit logic behind Bayesian views of weight decay and the like.

This is perhaps the most natural way of grafting Bayesian decision theory onto (weight-decay) BP. Unfortunately there are a number of odd aspects to this graft. To see this, first recall that O is the space of possible patterns on a neural net's output neurons. So in example (1) $o \in O$ is interpreted as a probability distribution of values y, given x (just like f), whereas in example (2) $o \in O$ is interpreted as a value of y. Since in example (2) we must assume that Y approximates a Euclidean vector space (see footnote (5)), in both examples O is a Euclidean vector (or at least a good approximation to such a vector) of dimension $|O|$. So in both examples, $\mathrm{net}(\mathbf{w}, x)$ is a Euclidean vector, and $\mathrm{net}(\mathbf{w}, .)$ is a mapping from X to such a vector.

Next recall that $\phi \in \Phi$ denotes a mapping from $x \in X$ to $o \in O$. Since X is finite, any ϕ is a vector of $|X|\,|O|$ real numbers. I will indicate the $|O|$ real numbers induced by ϕ for a particular input x by ϕ_x. Now $\mathrm{net}(\mathbf{w}, .)$ is a ϕ, and $\mathrm{net}(\mathbf{w}, x)$ is a ϕ_x. Moreover, for the usual reasons $\int d\mathbf{w}\ \mathrm{net}(\mathbf{w}, x)\ P(\mathbf{w} \mid t) = \int d\phi_x\ \phi_x\ P(\phi_x \mid t)$ (see appendix (1)). We can rewrite this as $\int d\phi\ \phi_x\ P(\phi \mid t)$. Therefore in either of our two examples $y^*(x, t)$ is determined by $\int d\phi\ \phi_x\ P(\phi \mid t)$. Accordingly, one might just as easily hope that y* is well-approximated if one uses the mode of $P(\phi \mid t)$ as hope that it is well-approximated if one uses (the ϕ associ-

---

pendent of σ_L. Intuitively, this means that our predictive distribution thinks that there is no noise in the generation of the test set, despite the presence of noise in the generation of the training set.

[7] Note that one might also try to use this kind of reasoning with the integral $\int d\phi_x\ \phi_x \times P(\phi_x \mid t)$. The idea would be to set $y^*(x, t)$ using the set of $|X|$ modes of $P(\phi_x \mid t)$ (one mode for each x). Intuitively, this correspond to setting $y^*(x, t)$ for each x to the mode of $P(y \mid x, t)$. Unfortunately, getting $P(\phi_x \mid t)$ from $P(\mathbf{w} \mid t)$ is more difficult than getting $P(\phi \mid t)$ from $P(\mathbf{w} \mid t)$ - for each x one must add a step of integrating over all $\phi_{x' \neq x}$: $p_{\Phi_x|T}(\phi_x \mid t) = \int d\phi'\ \delta(\phi'_x - \phi_x)\ p_{\Phi|T}(\phi' \mid t) = \int d\mathbf{w}\ \delta(\mathrm{net}(\mathbf{w}, x) - \phi_x)\ P(\mathbf{w} \mid t)$.

ated with) the mode of $P(\mathbf{w} \mid t)$.[7] However as demonstrated in the text, in general the MAP ϕ does ***not*** correspond to the ϕ associated with the MAP $\mathbf{w}$.

So should we try to find the MAP ϕ or the ϕ associated with the MAP $\mathbf{w}$? I.e., should we use BP modified to perform an ascent in $P(\phi \mid t)$, or should we use conventional BP which instead outputs the ϕ arising from an ascent of $P(\mathbf{w} \mid t)$? The proper way to address this question is to see which ϕ gives a better approximation to the Bayes-optimal $h_{x,y}$. However such a calculation is daunting. This is especially so when one realizes that the real question we want to answer is how the ϕ associated with a randomly chosen *local* peak of $P(\mathbf{w} \mid t)$ compares with a randomly chosen *local* peak of $P(\phi \mid t)$. (Gradient ascent procedures don't calculate global peaks in general.) So we have to resort to non-rigorous arguments to choose between the two ϕ's (and for that matter, to justify conventional BP within the Bayesian decision theory paradigm).

In most of the neural net literature, when language is used which draws a (potential) distinction between the two ϕ's, the goal is presented as finding the MAP ϕ. (E.g., (Denker and LeCun, 1991), (Weigend et al., 1992).) Indeed, most arguments for one or the other ϕ favor the MAP ϕ:

1) Regularized maximum likelihood techniques like spline fits can be viewed as finding the ϕ which best fits the data, subject to a penalty on ϕ (like its integrated curvature). Such techniques - which work very well in practice - are akin to finding the MAP ϕ, with $P(\phi)$ set by the "penalty on ϕ". (Note that usually there isn't even a parameter space $\mathbf{W}$ for these techniques - they're non-parametric.)

2) There are not unreasonable distributions $P(c \mid f, h, t)$ which say that one should find the MAP ϕ exactly. For example, the "catastrophe" $P(c \mid f, h, t)$ says that any deviation between h and f will be catastrophic: $P(c \mid f, h, t) = \delta(c + \delta(h - f))$. (A "catastrophe" corresponds to $c = 0$, no catastrophe to $c = -\infty$.) This means that $E(c \mid h, t) = -\int df\, P(f \mid t)\, \delta(h - f)$, so that the Bayes-optimal h is the MAP f, $\mathrm{argmax}_f\, P(f \mid t)$. In example (1), both $f_{x,y}$ and $\phi_{x,y}$ are $(|X| \times |Y|)$-dimensional Euclidean vectors, and they're related by the identity mapping. Accordingly, the MAP f is the MAP ϕ for that example.[8]

3) Simple utility: if you're going to use the MAP of a parameter of ϕ rather than of ϕ directly, you have to decide *which* parameterization scheme to use (and somehow justify it). In general, there's very little rigorous basis for such a decision.

4) One can argue that it makes sense to estimate a quantity as its MAP value simply because that's the most likely value of that quantity, given the data. This is the implicit

---

[8] However in example (2) $f_{x,y} = \frac{1}{\sigma\,(2\pi)^{1/2}}\, e^{-(\phi_x - y)^2 / 2\sigma^2}$, and $p_F[\, \frac{1}{\sigma\,(2\pi)^{1/2}}\, e^{-(\phi_x - y)^2 / 2\sigma^2}\,] \neq p_\Phi[\phi]$. For this case, to find the MAP f from the MAP ϕ we must invoke yet another correction term. In regard to this correction term, note that if K is the number of elements in O, then $f \in \mathbf{R}^{k|X|}$. Therefore f becomes infinite-dimensional as O = Y comes to approximate $\mathbf{R}^N$ (see footnote (5)), and extra care must be exercised in the analysis. However generically, when Y does approximate $\mathbf{R}^N$ the limits on y become infinite, and there's no way to distinguish values of our second correction term corresponding to different values of ϕ_x; the correction term is a constant, independent of ϕ_x. (As an illustrative example take $|X| = N = 1$, so the metric g_{ij} induced by the $\Phi \rightarrow F$ mapping is a 1×1

justification for MAP estimators in (Buntine and Weigend, 1991), for example. ("The posterior probabilities ... represent functions that should be maximized.") Since the quantity we directly want to know is (almost always) ϕ, not $\mathbf{w}$, such reasoning means we should guess the MAP ϕ rather than the ϕ associated with the MAP $\mathbf{w}$. Simply put, it makes intuitive sense to guess the most likely ϕ given the data, but it's very hard to come up with any intuition justifying use of the ϕ associated with the most likely $\mathbf{w}$.

5) Phrased differently, loss functions almost always concern ϕ, not (directly) $\mathbf{w}$. So the "natural" distribution of interest is $P(\phi \mid t)$, not $P(\mathbf{w} \mid t)$; $P(\phi \mid t)$ is more directly informative to us than is $P(\mathbf{w} \mid t)$. Therefore if one is going to use MAP estimators, the "natural" one is the MAP ϕ, not the MAP $\mathbf{w}$.

6) As an example, note that often we're interested in quantities like the posterior variance around the MAP estimate, which tells us how "peaked" the posterior distribution is about that estimate. Such variances are useful when they concern ϕ space, but variances in $\mathbf{w}$ space are not nearly so illuminating. So again, ϕ-space is the space of primary interest. (Even if we were to use the ϕ associated with the MAP $\mathbf{w}$, it's hard to see why we would be interested in the $\mathbf{w}$-space variance about that ϕ rather than the ϕ-space variance about it.) Indeed, see (Denker and LeCun, 1991; MacKay, 1992) for examples where people choose to estimate ϕ-space variances rather than $\mathbf{w}$-space variances.

Appendix Three: How to choose X.

This appendix extends appendix (2), by addressing the issue of what X to use.

One might consider having X be all possible patterns on the neural net's input neurons. However for many scenarios the resulting $|X|$ would be huge. (E.g., it's the number of values allowed by machine precision for representing real numbers.) This apparently makes calculating the Jacobian intractable in those scenarios, since it means that $|X|\ |O|$ is huge (though it's possible that there are some tricks which simplify calculation of the Jacobian). This would appear to make it impossible to apply this paper's results in those scenarios.

Fortunately though, we don't need to have X be the set of all possible input patterns. In fact, often we wouldn't want X to be all possible inputs even if this presented no calculational difficulties. This is due to the fact that we're using modes to approximate means (see appendix (2)). Usually the lower the dimension of a space the better the mean of a distribution across that space is approximated by that distribution's mode. Accordingly, it makes sense to approximate the vector $E(\Phi \mid t)_x = [\int d\phi\ \phi\ P(\phi \mid t)]_x = \int d\phi_x\ \phi_x\ P(\phi_x \mid t)$ by the mode of $P(\phi_x \mid t)$ rather than by the x-component of the mode of $P(\phi \mid t)$ (see footnote (7)).

In this alternative procedure, rather than finding the mode of $P(\phi \mid t)$ - a one-time only operation - and using that to guess outputs for all x's, instead for each new x one calculates the mode of $P(\phi_x \mid t)$. Obviously this procedure could be quite cpu-intensive (though less so than calculating $E(\Phi \mid t)_x$ for each new x, which is what we'd have to do to find the exact Bayes-optimal answer). However in practice the issue is mute, since as noted in footnote (7) we can't calculate $P(\phi_x \mid t)$, i.e., we can't marginalize down to a single x.

Fortunately there is an alternative that i) we can calculate, ii) has tractably small $|X|$, and iii) minimizes the need to recalculate distributions with each new x. This alternative is to marginalize down to a small set of x, X', for which $|\mathbf{W}| \le |X'|\ |O|$. We can perform such a marginalization simply by not considering any $x \notin X'$. (In particular, we would not consider such x when calculating the correction term.) This is because for generic $p_\Phi(\phi'_{x' \in X'}, \phi_{x \notin X'})$, marginalizing out all components of ϕ corresponding to $x \notin X'$ gives the distribution $p_{\Phi_{X'}}(\phi'_{x' \in X'})$, i.e., it's as though our input space were just X' all along.[9,10]

---

[9] Formally: To marginalize down to X' write $p_{\Phi_{X'}}(\phi'_{x'}) = \int d\phi_{x \notin X'}\ p_\Phi(\phi'_{x'}, \phi_x) = \int d\phi\ p_\Phi(\phi)\ \delta(\phi_{x'} - \phi'_{x'})$. In the usual way this equals $\int d\mathbf{w}\ p_{\mathbf{W}}(\mathbf{w})\ \delta(\mathrm{net}(\mathbf{w}, \cdot_{(X')}) - \phi'_{x'})$ (the delta being a multivariable delta function, and the subscripted dot notation indicating that we're viewing the net parameterized by $\mathbf{w}$ as a mapping from X' to O rather than from X to O). Accordingly, $p_{\Phi_{X'}}(\mathrm{net}(\mathbf{w}', \cdot_{(X')})) = \int d\mathbf{w}\ p_{\mathbf{W}}(\mathbf{w})\ \delta(\mathrm{net}(\mathbf{w}, \cdot_{(X')}) - \mathrm{net}(\mathbf{w}', \cdot_{(X')}))$, and similarly for conditional probabilities. Comparing to equation (1), we see that this is identical to what we would have gotten had X been restricted to X' from the start. QED.

[10] In other words, *as far as marginalization is concerned*, it doesn't matter how you define X (or X'); if you add elements to X, but then marginalize the resultant distributions down to the original set, you get the same distribution as you would if you'd never added those

In determining the effects of a particular choice of X' one should pay close attention to the issues discussed below equation (3). In general, since it is equivalent to marginalizing distributions involving the "full" X, a restriction on X doesn't introduce new singularities into the distributions (beyond those associated with the full X). On the other hand, it is possible that it removes singularities. However in general it won't remove the singularities discussed in section (3). For example, even with a restricted X, it's still true that if the weight leading out of a hidden neuron is zero one can vary all the weights leading into that neuron without affecting ϕ, and therefore you get a singularity.

One natural choice of X' is to have it involve T_X, the test set inputs. In such a case the training set inputs don't live in the same space as the inputs of ϕ's. So for example one couldn't simply write the likelihood as function+noise or some such. Instead it is given by $P(t \mid \phi'_{x'}) = \int d\phi_{x \notin X'} \, P(t \mid \phi'_{x'}, \phi_x) \, P(\phi_x \mid \phi_{X'})$, i.e., the restricted likelihood is an integral over the unrestricted likelihood. (Since we perform our calculations in **W**, and therefore in particular have as our likelihood P(t | **w**), we never explicitly encounter this complication.)

There's nothing blatantly wrong with having X' only partially overlap with T_X however, or perhaps even not overlap with T_X at all. (One could use X' to find a mode **w**, and then use that **w** to guess for all x, including the x in T_X.) Ultimately, the question is for what X' is the **w** produced by our procedure a better approximation to the Bayes-optimal $h_{x,y}$ for the elements of T_X. In general we can't perform the calculation to answer this question. Therefore we must rely on heuristics, just as in choosing between the MAP ϕ and the MAP **w** (see appendix (3)). The heuristics involved in that choice of MAP's suggest that X' should overlap with the test set maximally, but by no means prove it.

If we do have X' involve T_X but $|T_X| < |\mathbf{W}| / |O|$, then we have to choose some extra x's to go into X'. In contrast, for both reasons of computational tractability and to try to have our approximation as accurate as possible, we want |X'| to be as small as possible, and therefore we want to leave out elements of T_X if $|T_X| > |\mathbf{W}| / |O|$. (Under such a scheme, to make our guess for an $x'' \in T_X$ we'll have $X' \subset T_X$, where if we wish we can force things so that X' always includes x'' (this means that X' varies with x'').)

Because of this, even if we declare that X' should overlap with T_X, we still haven't fully specified the elements of X' when $|T_X| \neq |\mathbf{W}| / |O|$. And unfortunately, it isn't clear how to complete the specification. For example, if the elements of X' are close together, then we are coarsely approximating the case where X' consists of a single element, which we might expect to result in a better approximation to the Bayes-optimal guess. On the other hand, such an X' might also make the Jacobian ill-behaved. So it's not clear if the elements of X' should be close together or far apart. How best to choose X' is an area for future research.

Finally, note that to get |X| |O| small enough, as an alternative to restricting X one might restrict O. However this would mean denying the possibility of some outputs. Denying the possibility of some x's is fine, if they lie outside of t_X and T_X, so we know we can't encounter them. But denying the possibility of some o's is far more problematic. In gen-

---

extra elements. (What changes if one adds new elements are the *slices* of the distributions along certain x's, not the marginalizations.)

eral, one wouldn't expect it to result in a good approximation to the Bayes-optimal answer.

Appendix Four: Calculating the Hessian in ϕ-space.

This appendix presents a cursory overview of one way to calculate the Hessian in Φ of $\ln[\, p_{\Phi|T}(\phi \mid t)\,]$, i.e., of how to calculate the Hessian of $M'(\mathbf{w}, t)$ with respect to ϕ (see the end of section 4). To start restrict attention to a subregion of $\mathbf{W}$ and a subregion of Φ such that net($\mathbf{w}$, .) is one-to-one and onto over those subregions. This means in particular that $|\mathbf{W}| = |X|\, |O|$.

Write $\partial / \partial\phi_i = \Sigma_j\, [\partial\mathbf{w}_j / \partial\phi_i]\, \partial / \partial\mathbf{w}_j$. The difficulty with this expression is that we can't directly write down $\partial\mathbf{w}_j / \partial\phi_i$, only $\partial\phi_i / \partial\mathbf{w}_j$. However these two matrices are inverses of each other whether both are considered functions of ϕ or both are considered functions of $\mathbf{w}$. (This is because for any fixed i and j, $\Sigma_k\, [\partial\phi_i / \partial\mathbf{w}_k \times \partial\mathbf{w}_k / \partial\phi_j\,] = \delta(i, j)$, so long as the partial derivatives are all evaluated at the same point.) So for any function A(.) we have

$$(\partial / \partial\phi_i)\, A(\mathbf{w}(\phi)) = \Sigma_j\, \{[(\partial\phi / \partial\mathbf{w})^{-1}]_{i,j}\, (\partial / \partial\mathbf{w}_j)\, A(\mathbf{w})\},$$

where the inverse is understood to be of the whole matrix.

This partial derivative is a new function $B(\mathbf{w}(\phi))$. So recursing, we get

$$\begin{aligned}
&(\partial^2 / \partial\phi_i\partial\phi_j)\, A(\mathbf{w}(\phi)) \\
= \quad & \Sigma_k\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{i,k}\, (\partial / \partial\mathbf{w}_k)\, B(\mathbf{w}) \\
= \quad & \Sigma_k\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{i,k}\, (\partial / \partial\mathbf{w}_k)\, [\, \Sigma_l\, \{[(\partial\phi / \partial\mathbf{w})^{-1}]_{j,l}\, (\partial / \partial\mathbf{w}_l)\, A(\mathbf{w})\}\,] \\
= \quad & \Sigma_{k,l}\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{i,k}\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{j,l}\, (\partial^2 / \partial\mathbf{w}_k\partial\mathbf{w}_l)\, A(\mathbf{w}) \\
& \qquad + \Sigma_{k,l}\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{i,k}\, (\partial / \partial\mathbf{w}_l)\, A(\mathbf{w})\, (\partial / \partial\mathbf{w}_k)\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{j,l}.
\end{aligned}$$

Now we plug $A(\mathbf{w}) = \ln[\, p_{\Phi|T}(\text{net}(\mathbf{w}, .) \mid t)\,] = \ln[\, P_{\mathbf{W|T}}(\mathbf{w} \mid t)\,] - \ln[\, \|J_{\Phi,\mathbf{W}}(\mathbf{w})\|\,]$ into our two sums. The first sum acting on the first ln(.) will give a linear transformation of $I(\mathbf{w})$, where $I(\mathbf{w})$ is the Hessian of $M(\mathbf{w}, t)$ with respect to $\mathbf{w}$, i.e., $I(\mathbf{w})$ is the "Hessian" people usually calculate. (As an aside, note the close relationship between this linear transformation of $I(\mathbf{w})$ and the expression in equation (12) of (Buntine and Weigend, 1991).) The first sum acting on the second ln(.) gives the same linear transformation, but of the Hessian of the Jacobian. These linear transformations are straight-forward to evaluate.

For the second sum, we need to evaluate $(\partial / \partial\mathbf{w}_k)\, [(\partial\phi / \partial\mathbf{w})^{-1}]_{j,l}$. (The other factors are straight-forward to evaluate.) To do this, for expository simplicity define $C_{ij}(\mathbf{w}) \equiv \partial\phi_i / \partial\mathbf{w}_j$, and $D_{ij}(\mathbf{w}) \equiv \partial\mathbf{w}_i / \partial\phi_j = C^{-1}$. What we want to know is the gradient with respect to $\mathbf{w}$ of D.

We have (using implied sum convention) $C_{ij}\, D_{jk} = \delta_{ik}$ for all i, k, and (implicit) $\mathbf{w}$. Taking the gradient with respect to $\mathbf{w}$, and defining $E_{ij}(\mathbf{w}) \equiv \nabla_{\mathbf{w}}\, C_{ij}(\mathbf{w})$ and $F_{ij}(\mathbf{w}) \equiv \nabla_{\mathbf{w}}\, D_{ij}(\mathbf{w})$, we have $0 = E_{ij}\, D_{jk} + C_{ij}\, F_{jk}$ for all i, k, and (implicit) $\mathbf{w}$. Multiplying on the left by $[C^{-1}]_{li} = D_{li}$, we get $0 = D_{li}\, E_{ij}\, D_{jk} + F_{lk}$. So for all l, k, and (implicit) $\mathbf{w}$, F_{lk} - which is nothing other than that second term we want to calculate - equals $-(D_{li}\, E_{ij}\, D_{jk})$.

In this way we can evaluate both sums at any particular point $\mathbf{w}$, and therefore calculate the Hessian of $M'(\mathbf{w}, t) = \ln[\ p_{\Phi|\mathbf{T}}(net(\mathbf{w}, .) \mid t)\ ]$ with respect to ϕ at any point $\mathbf{w}$.

When $|\mathbf{W}| \neq |X|\ |O|$ things are a bit more complicated. If $|\mathbf{W}| > |X|\ |O|$, the probability density at a particular ϕ is determined by the density at a set of $\mathbf{w}$'s that extends globally in $\mathbf{W}$, beyond infinitesimal neighborhoods of any particular $\mathbf{w}'$. Accordingly, local characteristics of the density over Φ (e.g., the Hessian of $\ln[\ p_{\Phi|\mathbf{T}}(\phi \mid t)\ ]$ with respect to ϕ) correspond to global characteristics in $\mathbf{W}$, and such local characteristics of the density over Φ can not be expressed in terms of properties of a single point in $\mathbf{W}$. In other words, one has no choice but to (try to) perform the required integration over $\mathbf{w}$ (see section (4) of the text), and then take partial derivatives with respect to ϕ.

If $|\mathbf{W}| < |X|\ |O|$ things aren't so bad; we can use the analysis of the case where $|\mathbf{W}| = |X|\ |O|$, if we first perform a rotation of Φ. More precisely: Let the point at which we wish to know the Hessian be $\phi' \equiv net(\mathbf{w}', .)$. Calculate the $|\mathbf{W}|$ Φ-space vectors $\partial\phi\ /\ \partial\mathbf{w}_i$, one vector for each $i \in \{1, 2, ..., |\mathbf{W}|\}$. These vectors give the tangent plane to $S(\mathbf{W})$ (see section (4) of the text) at ϕ'. Now rotate Φ so that the first $|\mathbf{W}|$ of the coordinates of Φ are parallel to this tangent plane. Refer to this rotated version of Φ as Φ_R. Note that probability densities over Φ_R equal densities over Φ (up to a rotation), since the determinant of a rotation matrix equals 1 (and therefore the "correction term" for such a rotation equals 1). Refer to the space of those first $|\mathbf{W}|$ coordinates of Φ_R as Φ'_R. Note that $|\Phi'_R| = |\mathbf{W}| < |\Phi|$.

Having performed the rotation, we can examine the Hessian with respect to Φ'_R (i.e., with respect to coordinates $[\Phi_R]_{i \leq |\mathbf{W}|}$) rather than the Hessian with respect to Φ. To do this simply perform the same calculation as at the beginning of this appendix, with Φ'_R substituted for Φ throughout, and in particular substituted for Φ in the derivatives with respect to $\mathbf{w}$. (Note that partial derivatives with respect to $[\Phi_R]_{i > |\mathbf{W}|}$ are all infinite, since our density equal zero off of $S(\mathbf{W})$.)

Acknowledgements

Thanks to David Rosen and Wray Buntine for stimulating discussion, and to TXN Inc. and the SFI for funding.

References

Berger, J. (1985). *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag.

Bridle, J. (1989). Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In F. Fougelman-Soulie and J. Herault (Eds.), *Neuro-computing: Algorithms, architectures, and applications*. Springer-Verlag.

Buntine, W., Weigend, A. (1991). Bayesian back-propagation. *Complex Systems*, **5**, p. 603.

Denker, J., LeCun, Y. (1991). Transforming neural-net output levels to probability distributions. In *Neural Information Processing Systems 3*, R. Lippman et al. (Eds).

Fefferman, C. (1993). Reconstructing a neural net from its output. Sarnoff Research Center TR 93-01.

Hassibi, B., and Stork, D. (1992). Second order derivatives for network pruning: optimal brain surgeon. Ricoh Tech Report CRC-TR-9214.

MacKay, D. (1992). Bayesian Interpolation, *and* A Practical Framework for Backpropagation Networks. *Neural Computation*, **4**, pp. 415 and 448.

Neal, R. (1993). Bayesian learning via stochastic dynamics. In *Neural Information Processing Systems 5,* S. Hanson et al. (Eds). Morgan Kaufmann.

Nowlan, S., and Hinton, G. (1994). Simplifying Neural Networks by Soft Weight-Sharing. In *Theories of Induction: Proceedings of the SFI/CNLS Workshop on Formal Approaches to Supervised Learning*, D. Wolpert (Ed.). Addison-Wesley, to appear.

Rissanen, J. (1986). Stochastic complexity and modeling. *Ann. Stat.*, **14**, p. 1080.

Rosen, D. (1994). Scoring the forecaster by mean resulting payoff of a distribution of decision problems.To appear in *Maximum entropy and Bayesian methods*, G. Heidbreder (Ed.), Kluwer.

Titterington, D., et al. (1985). Statistical analysis of finite mixture distributions. Wiley.

Weigend et al., (1992). Predicting susnspots and exchange rates with connectionist networks. In *Nonlinear modeling and forecasting*, M. Casdagli and S. Eubank (Eds.), Addison-Wesley.

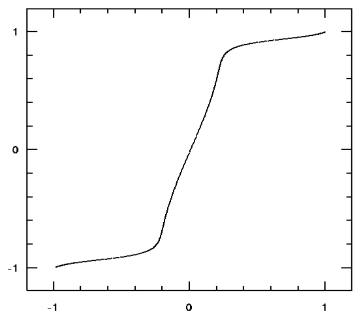
Wolpert, D., (1992). On the connection between in-sampling testing and generalization error. *Complex Systems,* **6**, p. 47.

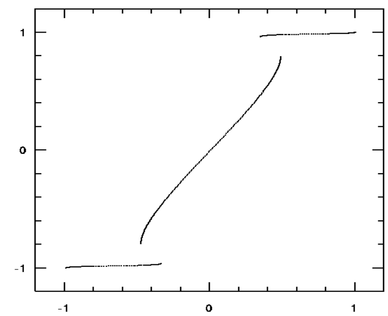
Wolpert, D. (1993). On the use of evidence in neural networks. In *Neural Information Processing Systems 5*, S. Hanson et al. (Eds). Morgan-Kauffman.

Wolpert, D. (1994). Filter likelihoods and exhaustive learning. To appear in *Compuational Learning Theory and Natural Learning Systems: Volume II Natural Learning Systems*, S. Hanson et al. (Ed.'s). MIT Press.

Wolpert, D., Strauss, C. (1994). What Bayes has to say about the evidence procedure. To appear in *Maximum entropy and Bayesian methods*, G. Heidbreder (Ed.), Kluwer.

FIGURES





Figures 1 through 3: Train using unmodified BP on training set t, and feed input x into the resultant net. The horizontal axis gives the output you get. If t and x were still used but training had been with modified BP, the output would have been the value on the vertical axis. In succession, the three figures have α = .6, .4, .4, and m = 1, 4, 1.

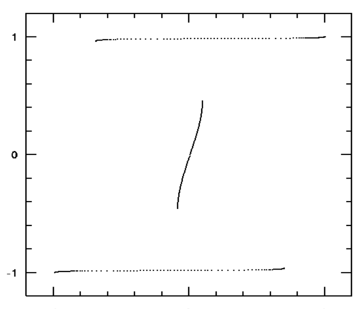


Figure 3.

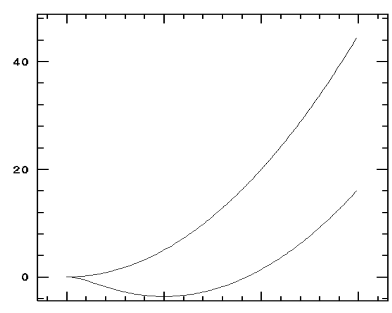


Figure 4: The horizontal axis is $|\mathbf{w}_i|$. The top curve depicts the weight decay regularizer, $\alpha \mathbf{w}_i^2$, and the bottom curve shows that regularizer modified by the correction term. α = .2.