International Journal of Neural Systems, Vol. 8, Nos. 5 & 6 (October/December, 1997) 489–498
© World Scientific Publishing Company

UNIVERSAL DISTRIBUTION OF SALIENCIES FOR PRUNING IN LAYERED NEURAL NETWORKS

J. GORODKIN[*†¶], L. K. HANSEN[‡‖], B. LAUTRUP[*††] and S. A. SOLLA[*§]
[*] *CONNECT, The Niels Bohr Institute, Blegdamsvej 17, 2100 Copenhagen Ø, Denmark*
[†] *Center for Biological Sequence Analysis, The Technical University of Denmark, Building 206, 2800 Lyngby, Denmark*
[‡] *CONNECT, Department of Mathematical Modelling, The Technical University of Denmark, Building 305, 2800 Lyngby, Denmark*

Received June 30, 1997
Accepted July 31, 1997

A better understanding of pruning methods based on a ranking of weights according to their *saliency* in a trained network requires further information on the statistical properties of such saliencies. We focus on two-layer networks with either a linear or nonlinear output unit, and obtain analytic expressions for the distribution of saliencies and their logarithms. Our results reveal unexpected universal properties of the log-saliency distribution and suggest a novel algorithm for saliency-based weight ranking that avoids the numerical cost of second derivative evaluations.

1. Introduction

The problem of supervised learning in layered neural networks is a two stage process. A choice of architecture leads to the implicit definition of an associated parameter space $\{\mathbf{w}\}$, which represents the ensemble of weights whose values need to be determined in order to fully specify the network. This parameter space is then searched so as to identify specific parameter values $\mathbf{w}^*$. The goal is to obtain a network with low *generalization error* E_G, a quantity that measures the difference between the input–output map implemented by the network and the target map.

The training of feed-forward networks is usually formulated as an optimization problem: values for the parameters $\mathbf{w}$ are chosen so as to minimize a *learning error* E_L, defined as a sum over a set of training examples given in the form of input–output pairs. The extent to which low learning error results in low generalization error is controlled by the prior choice of network architecture; the possibility of using the information provided by the data to guide this choice has been explored in a variety of learning algorithms.[1]

Here we focus on the family of *pruning* algorithms, based on the elimination of redundant weights and/or neurons during the training process. A variety of such methods has been introduced in recent years, some based on the removal of neurons[2–4] and some on the removal of individual weights.[5–9] The goal in either case is to control the size of the network so as to obtain the smallest possible network compatible with learning the training set. Capacity arguments[10] indicate that improved generalization should result from

[§]Permanent address: Department of Physiology and Institute for Neuroscience, Northwestern University Medical School, Chicago, IL 60611 and Department of Physics and Astronomy, Northwestern University, Evanston, IL 60208, USA.
E-mail: solla@nbi.dk
[¶]E-mail: gorodkin@cbs.dtu.dk
[‖]E-mail: lkhansen@ei.dtu.dk
[††]E-mail: lautrup@connect.nbi.dk

this reduction in network size; for numerical experiments that confirm this prediction see for example Ref. 11. Trained networks of minimal size that implement a target map are also of interest as a potential tool for comparing the intrinsic complexity of different tasks, and often provide an interpretable rendition of the computational strategy through which the map is implementable in a neural network representation.[12]

We consider a weight pruning scheme summarized as follows: the network is trained to a minimum of the learning error, the weight that would cause the smallest increase in learning error if removed is identified and removed, and the resulting smaller network is retrained to reach a new minimum of the learning error.[5,6,8,9] We follow the approach of Le Cun *et al.*[5], who propose a second order estimate of the *saliency* of each individual weight, defined as the increase in learning error that would result from its removal. Our goal is to characterize the expected distribution of saliencies over the weights of a two-layer network, an architecture that has been shown to provide a universal approximator for the implementation of functions from an N-dimensional input space onto a scalar output.[13,14]

General assumptions about the statistical properties of the input data and the weights of the trained network allow us to calculate the distribution of the logarithm of the saliencies, to find an unexpected result: that the distribution is universal except for a translation that contains all dependence on the training data. We present the analytic derivation of this result for two fundamental types of two-layer networks[15]: one with a linear output unit and trained through a quadratic error function, and another one with a sigmoidal output unit and trained through a logarithmic error function of the Kullback–Leibler type. Results for a single-layer linear network are included for comparison. Numerical simulations are used to illustrate the validity of our assumptions about the statistical distribution of the weights in a trained network, and to verify our predictions about the universal form of the saliency distribution.

An intriguing consequence of our calculations is a novel pruning algorithm that partially justifies and easily extends the simplest form of pruning in which the saliency of a weight is assumed to be determined only by its magnitude.

2. The Saliencies

The saliency of a weight in a layered neural network is defined as the increase in learning error that would result from its removal. Le Cun *et al.*[5] have proposed a second order method to estimate this increase for networks that have been trained to a minimum of the learning error.

The first derivative of the learning error E_L is zero at the minimum; the dominant contribution to the saliency thus comes from the second derivatives. Higher order contributions are neglected, and the assumption that only one weight is removed at a time is used to neglect the off-diagonal terms in the matrix of second derivatives, to obtain

$$\delta E_L^k \approx \frac{1}{2} \frac{\partial^2 E_L}{\partial w_k^2}\bigg|_{\mathbf{w}^*} w_k^2 \tag{1}$$

for the increase in learning error associated with the removal of the kth weight. The *saliency* s_k of the kth weight is defined as this increase: $s_k \equiv \delta E_L^k$, and it is in this approximation fully determined by the magnitude of the weight w_k and the corresponding diagonal element of the Hessian matrix of second derivatives of the learning error evaluated at the minimum: $\mathbf{w} = \mathbf{w}^*$.

We are interested in layered networks that implement maps from an N-dimensional input onto a scalar output; we thus restrict ourselves to two-layer networks with a single hidden layer of sigmoidal units connected to one output unit, which have been shown to be universal approximators for the implementation of such functions.[13,14] We derive results for a linear output unit and present their extension to the case of a nonlinear output unit. Results for a single-layer linear network are included for comparison.

The output O^μ of a two-layer network with a linear output unit under presentation of input $\mathbf{x}^\mu$ is given by:

$$O^\mu = \sum_{i=1}^{K} W_i \tanh\left(\sum_{j=0}^{N} w_{ij} x_j^\mu\right) - W_0\,, \tag{2}$$

where $x_0^\mu = -1$ for all μ, w_{i0} is the threshold of the ith hidden unit, and W_0 is the threshold of the output unit. The quantity w_{ij} refers to the weight from the jth input unit to the ith hidden unit and

W_i refers to the weight from the ith hidden unit to the output unit. The number of inputs is N and the number of hidden units is K. The equivalent expression for the case of a nonlinear output unit is:

$$O^\mu = \tanh\left(\sum_{i=1}^{K} W_i \tanh\left(\sum_{j=0}^{N} w_{ij} x_j^\mu\right) - W_0\right). \tag{3}$$

We concentrate here on the saliencies for the input-to-hidden weights; the saliencies of the hidden-to-output weights are easily found following a similar procedure.[16] For a linear output unit we use a quadratic error function $E_L = \frac{1}{2p}\sum_{\mu=1}^{p}(y^\mu - O^\mu)^2$ to measure the distance between target outputs y^μ and actual outputs O^μ. The corresponding saliencies take the form[11]:

$$s_{ij} = \frac{1}{2} w_{ij}^2 W_i^2 \frac{1}{p} \sum_{\mu=1}^{p} [1 - \tanh^2(h_i^\mu)]^2 (x_j^\mu)^2 , \tag{4}$$

where the notation $h_i^\mu = \sum_{j=0}^{N} w_{ij} x_j^\mu$ is introduced to indicate the activation of the ith hidden unit under presentation of the μth example. For a nonlinear output unit we use the Kullback–Leibler entropy[17] as the error function; for output units confined to the $[-1, +1]$ interval through a nonlinearity of the "tanh" type, the error is written as: $E_L = \frac{1}{2p}\sum_{\mu=1}^{p}$ $[(1+y^\mu)\log\frac{1+y^\mu}{1+O^\mu} + (1-y^\mu)\log\frac{1-y^\mu}{1-O^\mu}]$. The corresponding saliencies take the form[16]:

$$s_{ij} = \frac{1}{2} w_{ij}^2 W_i^2 \frac{1}{p} \times \sum_{\mu=1}^{p} [1 - \tanh^2(H^\mu)][1 - \tanh^2(h_i^\mu)]^2 (x_j^\mu)^2 , \tag{5}$$

where H^μ is the activation of the output unit under the presentation of the μth example, a quantity equal to the output O^μ of the network with a linear output unit, Eq. (2).

It is the logarithm of these saliencies that we now evaluate.

3. The Distribution of Saliencies and Their Logarithms

Our analysis of the statistical properties of the saliencies of a trained layered network is based on simple assumptions about the statistical properties of the corresponding weights.

A simultaneous sign change of all weights in either network (2) or network (3) leaves the output invariant; this symmetry leads to the assumption that the probability distribution for individual weights is symmetric around zero and has zero mean. We choose a Gaussian approximation to the distribution of weights in the trained network. This further assumption is well supported by numerical evidence obtained through the training of layered networks architecturally too large for the implementation of the target map. Consider the shape of the surface $E_L(\mathbf{w})$ near the minimum at $\mathbf{w} = \mathbf{w}^\star$ for such a network: the surface remains essentially flat in a significant interval around $\mathbf{w}^\star$ along those directions corresponding to redundant parameters. Gradient-descent weight updates in the vicinity of the minimum thus result in stochastic variations, which are almost uncorrelated from one time step to the next and add up to weights that are normally distributed. Numerical experiments support this picture and indicate that the Gaussian character of the weight distribution increases with increasing learning time in the vicinity of the minimum.

If the weight distribution has finite covariance matrix, and the individual components x_j^μ of the p input patterns are independently drawn from a distribution $P(x)$, the *central limit theorem* can be invoked to argue that the activation h_i^μ of the ith hidden unit under presentation of the μth pattern is a normally distributed random variable with zero mean and variance $\sigma_h^2 = (N+1)\sigma_w^2\sigma_x^2$, where σ_w^2 is the variance of the input-to-hidden weights and σ_x^2 stands for $(N+1)^{-1}\sum_{j=0}^{N}(x_j^\mu)^2$, independent of μ. Correlations $\langle h_i^\mu x_j^\mu \rangle$ vanish for weights w_{ij} that are stochastic variables with zero mean.

We now concentrate on the case of a two-layer network with a linear output unit and compute the saliencies in Eq. (4) and the distribution of their logarithms. We have applied similar arguments to analyze the case of a two-layer network with a nonlinear output unit; the corresponding results are summarized as we go along.

In the case of a linear output unit, consider the quantity

$$Q = \frac{1}{2p}\sum_{\mu=1}^{p}[1 - \tanh^2(h^\mu)]^2 (x^\mu)^2 . \tag{6}$$

The statistical properties of this stochastic variable are independent of i and j; the fluctuations of this single stochastic variable assign different values of Q to different input-to-hidden weights. It is useful to write $Q = (1/p)\sum_{\mu=1}^{p} v^\mu$, and note that different terms v^μ are uncorrelated, as the statistical independence of the input patterns guarantees that both the x's and the h's are uncorrelated from example to example. It then follows that $\langle Q\rangle = \langle v\rangle$ and $\sigma_Q^2 = (1/p)\sigma_v^2$. The relative variance

$$\frac{\sigma_Q}{\langle Q\rangle} \sim \frac{1}{\sqrt{p}} \to 0 \quad \text{as} \quad p \to \infty\,. \tag{7}$$

As the number p of examples grows the fluctuations of Q around its mean $\langle Q\rangle$ become negligible; in this limit the stochastic variable Q becomes *self-averaging* and it can be replaced by $\langle Q\rangle$. The saliencies in Eq. (4) can then be written as

$$s_{ij} = w_{ij}^2 W_i^2 \langle Q\rangle\,. \tag{8}$$

The same argument applies to the case of a nonlinear output unit; it suffices to redefine

$$Q = \frac{1}{2p}\sum_{\mu=1}^{p}[1-\tanh^2(H^\mu)][1-\tanh^2(h^\mu)]^2(x^\mu)^2\,, \tag{9}$$

to show that the saliencies in Eq. (5) can also be written in the form of Eq. (8).

Equation (8) thus provides a compact expression for the input-to-hidden saliencies s_{ij} for two-layer networks with either a linear or nonlinear output unit. Since the value of $\langle Q\rangle$ is independent of i and j, all the information needed to rank these weights by order of increasing saliency is contained in the product of the magnitude of two weights: w_{ij} and W_i. This result reveals a simple way of implementing OBD in a two-layer network; it improves upon simple pruning schemes based on ranking weights according to only their own magnitude, and also avoids the numerical cost associated with the computation of second derivatives.

Hidden-to-output saliencies s_i, labeled only by the index i of the corresponding weight W_i, are easily shown[16] to be proportional to W_i^2, with a self-averaging proportionality factor that is independent of i. Pruning according to a saliency-based ranking thus reduces for the hidden-to-output weights of a two-layer network to a simple magnitude-based pruning scheme.

The evaluation of the saliencies in Eq. (8) requires the computation of the parameter $\langle Q\rangle = \int_{-\infty}^{\infty} dQ\; Q\; P(Q)$, which contains explicit information about the data. In the case of a linear output unit, the pertinent distribution is

$$P(Q) = \int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dh^\mu dx^\mu P(h^\mu)P(x^\mu) \times \delta\left(Q - \frac{1}{2p}\sum_{\mu=1}^{p}[1-\tanh^2(h^\mu)]^2(x^\mu)^2\right), \tag{10}$$

with

$$P(h) = \frac{1}{\sqrt{2\pi}\sigma_h}\exp\left[-\frac{h^2}{2\sigma_h^2}\right]\,. \tag{11}$$

The corresponding value of $\langle Q\rangle$ can be easily calculated in the regimes $\sigma_h^2 \gg 1$ and $\sigma_h^2 \ll 1$ (see Appendix A), to obtain $(2\sigma_x^2)/(3\sqrt{2\pi}\sigma_h)$ and $\sigma_x^2/2$, respectively.

A similar analysis yields an expression for $\langle Q\rangle$ in the case of a nonlinear output unit (see Appendix A); it depends not only on the variance σ_h for the activation of the hidden units but also on the variance σ_H for the activation of the output unit. Asymptotic results are $(2\sigma_x^2)/(3\pi\sigma_h\sigma_H)$ in the large variance limit and $\sigma_x^2/2$ in the small variance limit.

We now turn to calculating the distribution for the logarithm of the saliencies; the reason for focusing on the logarithm of the saliencies rather than the saliencies themselves will become clear in the process. Consider the logarithm of the saliencies in Eq. (8)

$$z = \log s = \log w^2 W^2 \langle Q\rangle\,, \tag{12}$$

where indices ij identifying a specific input-to-hidden weight are omitted for simplicity. We are interested in the distribution

$$P(z) = \int_{-\infty}^{\infty} dw \int_{-\infty}^{\infty} dW \times \delta(z - \log w^2 W^2\langle Q\rangle)P_w(w)P_W(W)\,, \tag{13}$$

with P_W and P_w given by

$$\begin{aligned} P_w(w) &= \frac{\exp[-w^2/(2\sigma_w^2)]}{\sqrt{2\pi}\sigma_w}\,,\\ P_W(W) &= \frac{\exp[-W^2/(2\sigma_W^2)]}{\sqrt{2\pi}\sigma_W}\,. \end{aligned} \tag{14}$$

The resulting Gaussian integrals can be performed (see Appendix B) to obtain[16]:

$$P(z) = \frac{\alpha(z)}{\pi}K_0(\alpha(z))\,, \tag{15}$$

where K_0 is the modified Bessel function of the second kind of order zero and

$$\alpha(z) = \sqrt{\frac{e^z}{\langle Q\rangle}}\,\frac{1}{\sigma_w\sigma_W} = \sqrt{\exp(z - \log\sigma_w^2\sigma_W^2\langle Q\rangle)}\,. \tag{16}$$

This result follows from Eq. (8) and is thus valid for a two-layer network with either a linear or nonlinear output unit; the character of the output unit simply selects the appropriate form for $\langle Q\rangle$, as discussed in Appendix A.

Note that a change in $\langle Q\rangle$ only contributes in a translation along the z-axis; the distribution for the logarithm of the saliencies has a shape which is *universal* in that it is independent of the data and thus of the task that the network is being trained for! The shape of the distribution is shown in Fig. 1.

The distribution of saliencies can be found through a similar procedure, to obtain[16]:

$$P(s) = \alpha_1(s)K_0(\alpha_2(s))\,, \tag{17}$$

with $\alpha_1(s) = [2\pi\sigma_w^2\sigma_W^2\sqrt{\langle Q\rangle s}]^{-1}$ and $\alpha_2(s) = (\sigma_W/\sigma_w)\sqrt{s/\langle Q\rangle}$. Note that a change in $\langle Q\rangle$ results in a change of the actual shape of the saliency distribution.

For comparison we quote here the corresponding results for a single-layer linear network with no hidden units. A similar but simpler calculation[16] leads to results for the distribution of the saliencies s and their logarithms z:

$$P(z) = \frac{\alpha(z)}{\sqrt{\pi}}\exp(-\alpha^2(z))\,, \tag{18}$$

with

$$\alpha(z) = \sqrt{\frac{e^z}{\sigma_x^2}}\,\frac{1}{\sigma_w} = \sqrt{\exp(z - \log\sigma_x^2\sigma_w^2)}\,, \tag{19}$$

and

$$P(s) = \alpha_1(s)\exp(-\alpha_2(s))\,, \tag{20}$$

with $\alpha_1(s) = [\sigma_w\sqrt{\pi s}]^{-1}$ and $\alpha_2(s) = (1/\sigma_w)\sqrt{s}/2$. For this simple architecture it is necessary to consider the possibility $\langle w\rangle \neq 0$; the corresponding calculations can be carried through,[16] and the results reveal a scaling of both height and width of the distribution $P(s)$ with $\langle w\rangle$.

4. Numerical results

We now present numerical experiments on two-layer networks that justify our assumption of zero-mean normally distributed trained weights.[16] The saliencies s associated with the various weights are computed for the trained network following the original prescription by Le Cun *et al.* as summarized in Eq. (1). The corresponding distribution

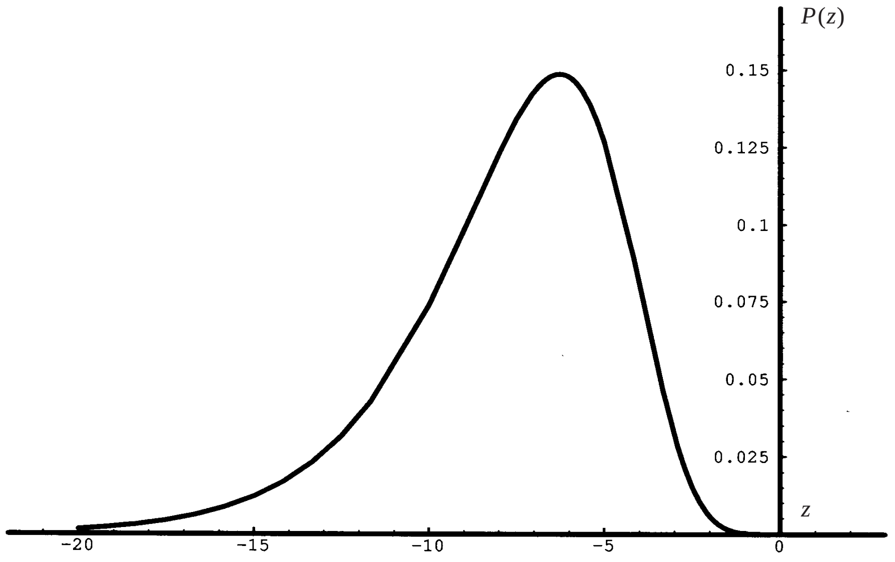

Fig. 1. Distribution of the logarithm of the saliencies z for the input-to-hidden weights of a two-layer network with a linear output unit. $P(z)$ is shown for $\sigma_h \ll 1$, with $\sigma_w = \sigma_W = 0.2$ and $\sigma_x = 1$, as is the case for ± 1 binary input components.

for the logarithms $z = \log s$ of the saliencies is found to be in good agreement with our theoretical predictions.

Numerical experiments reported here are for the *contiguity problem*,[18,19] in which strings of N binary components ± 1 are classified into categories according to the number of contiguous clumps of $+1$'s present in the pattern. The problem is simplified into a dichotomy by focusing on patterns that contain either only two such clumps (to be mapped onto an output of -1) or three such clumps (to be mapped onto an output of $+1$). Here

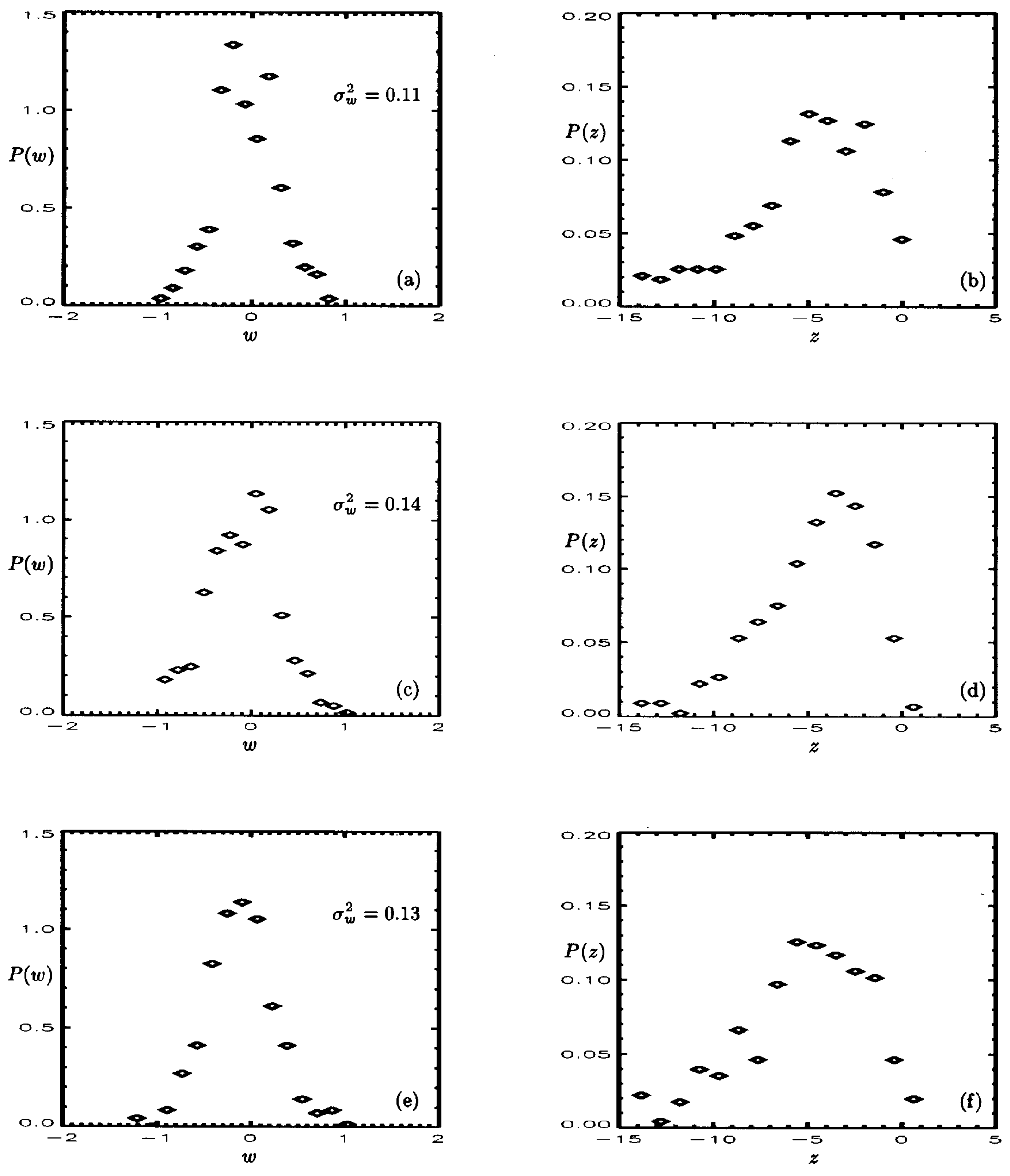

Fig. 2. The weight distribution $P(w)$ and log-saliency distribution $P(z)$ for the input-to-hidden weights of a two-layer neural network with $N = 10$ input units and $K = 40$ hidden units trained on the contiguity problem. Histograms based on data for all 400 input-to-hidden weights are shown 10 iterations after reaching the minimum of the learning error [(a) and (b)], 4000 iterations after reaching the minimum [(c) and (d)], and 20000 iterations after reaching the minimum [(e) and (f)]. The variance σ_w^2 of the trained weight distribution is indicated in each case.

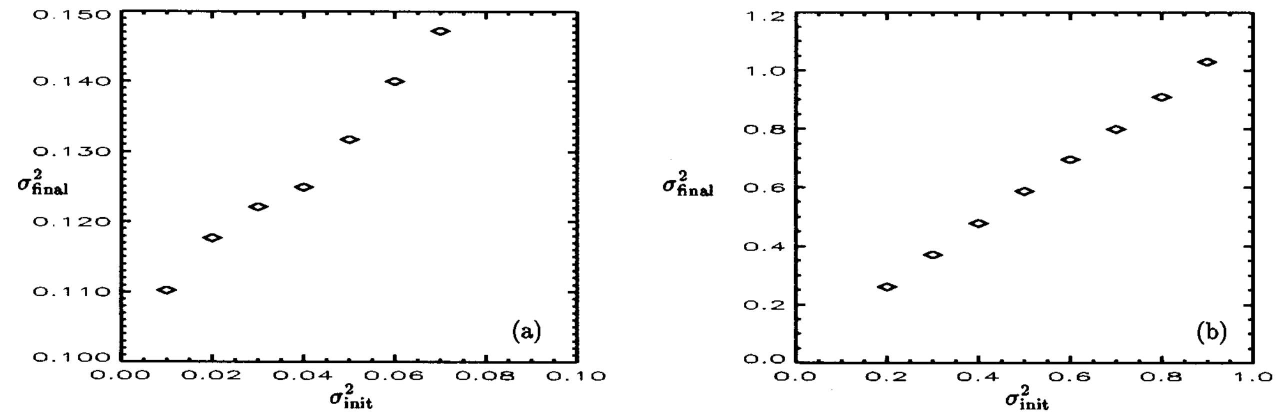

Fig. 3. The variance of the final weight distribution as a function of the variance of the initial weight distribution in two regimes: (a) $\sigma^2_{\text{init}} \in [0, 0.1]$ and (b) $\sigma^2_{\text{init}} \in [0, 1]$. Data for the trained network has been taken 1000 iterations after reaching the minimum of the learning error.

we consider $N = 10$; out of a total of 1024 possible patterns we only consider 792, of which 330 belong to the two-clump category and 462 to the three-clump category.

We randomly select a training set of size $p = 79$ and use it to train a two-layer nonlinear network as described by Eq. (3), with $K = 40$ hidden units and 481 parameters to be determined by training. Learning proceeds by gradient-descent on an error function of the Kullback–Leibler type. Target values are chosen at ± 0.9 instead of ± 1 to avoid saturation effects that conspire against the validity of the second order approximation of Eq. (1).

Numerical results are shown in Fig. 2 for a training session in which weights were initially drawn from a uniform distribution in the interval $[-1/\sqrt{N}, 1/\sqrt{N}]$, with $\langle w \rangle = 0$ and $\sigma^2_w = 0.033$. The weight distribution $P(w)$ and log-saliency distribution $P(z)$ were measured upon reaching the minimum of the learning error [Figs. 2(a) and 2(b)], after 4000 additional iterations of the gradient-descent algorithm [Figs. 2(c) and 2(d)], and again after 20000 additional iterations [Figs. 2(e) and 2(f)]. No weights were pruned; the histograms shown in Fig. 2 gather the information contained in all 400 input-to-hidden weights.

Similar results are obtained if the initial weights are drawn from a zero-mean Gaussian as opposed to a uniform distribution.[16] Numerical results shown in Fig. 3 reveal a linear correlation between the initial and final values of σ^2_w.

5. Summary

Simple assumptions about the statistical properties of weights in a trained two-layer network are supported by numerical evidence and used here to predict the expected distribution of saliencies. Ranking of weights according to their post-training saliency is a crucial ingredient of pruning algorithms such as Optimal Brain Damage; for the case of a two-layer network our results provide a simple algorithm for implementing this prescription without the computational cost associated with second-derivative evaluations. An unexpected outcome of our calculations is the finding of a universal shape for the distribution of the logarithms of the saliencies. The discovery of a task-independent profile for $P(z)$ leaves open the question of how to utilize the information contained in the data to formulate a stopping criterion for the pruning process.

Acknowledgments

We thank Anders Krogh, Claus Svarer, and Ole Winther for useful discussions. This research was supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center (CONNECT) and the Danish National Research Foundation through the Danish Center for Biological Sequence Analysis.

Appendix

A. The Mean $\langle Q\rangle$

We now calculate $\langle Q\rangle = \int_{-\infty}^{\infty} dQQP(Q)$, as needed to evaluate the saliencies in Eq. (8). In the case of a linear output unit:

$$\begin{aligned}\langle Q\rangle &= \int_{-\infty}^{\infty} dQQ \int_{-\infty}^{\infty} \prod_{\mu=1}^{p} dh^\mu dx^\mu P(h^\mu)P(x^\mu)\,\delta\left(Q - \frac{1}{2p}\sum_{\mu=1}^{p}[1-\tanh^2(h^\mu)]^2(x^\mu)^2\right)\\ &= \int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dx^\mu P(x^\mu)\int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dh^\mu P(h^\mu)\int_{-\infty}^{\infty} dQQ\,\delta\left(Q-\frac{1}{2p}\sum_{\mu=1}^{p}[1-\tanh^2(h^\mu)]^2(x^\mu)^2\right)\\ &= \int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dx^\mu P(x^\mu)\int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dh^\mu P(h^\mu)\frac{1}{2p}\sum_{\nu=1}^{p}[1-\tanh^2(h^\nu)]^2(x^\nu)^2 \qquad (21)\\ &= \frac{1}{2p}\int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dx^\mu P(x^\mu)\sum_{\nu=1}^{p}(x^\nu)^2\int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dh^\mu P(h^\mu)[1-\tanh^2(h^\nu)]^2\\ &= \frac{1}{2}\sigma_x^2\int_{-\infty}^{\infty}\frac{dh}{\sqrt{2\pi}\sigma_h}\mathrm{e}^{-\frac{1}{2}\frac{h^2}{\sigma_h^2}}[1-\tanh^2(h)]^2\,,\end{aligned}$$

with $\sigma_x^2 = \int_{-\infty}^{\infty} dxP(x)x^2$, as before.

This integral is easily evaluated in two limits. For $\sigma_h^2 \gg 1$, expand $\exp[-\frac{1}{2}\frac{h^2}{\sigma_h^2}] \approx 1 - h^2/(2\sigma_h^2)$, and use $1-\tanh^2(h) = d\tanh(h)/dh$ to obtain

$$\int_{-\infty}^{\infty}\frac{dh}{\sqrt{2\pi}\sigma_h}[1-\tanh^2(h)]\frac{d\tanh(h)}{dh} = \frac{1}{\sqrt{2\pi}\sigma_h}\int_{-1}^{1}d\tilde{h}(1-\tilde{h}^2) = \frac{4}{3}\frac{1}{\sqrt{2\pi}\sigma_h}\,, \qquad (22)$$

and

$$\langle Q\rangle = \frac{2}{3}\frac{\sigma_x^2}{\sqrt{2\pi}\sigma_h} + \mathcal{O}((\sigma_h)^{-3})\,. \qquad (23)$$

For $\sigma_h \ll 1$, substitute $\tilde{h} = h/\sigma_h$, to obtain

$$\int_{-\infty}^{\infty}\frac{d\tilde{h}}{\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\tilde{h}^2}[1-\tanh^2(\tilde{h}\sigma_h)]^2 \approx \int_{-\infty}^{\infty}\frac{d\tilde{h}}{\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\tilde{h}^2} = 1\,, \qquad (24)$$

and

$$\langle Q\rangle = \frac{1}{2}\sigma_x^2\,. \qquad (25)$$

In the case of a nonlinear output unit,

$$\begin{aligned}\langle Q\rangle = \int_{-\infty}^{\infty} dQQ\int_{-\infty}^{\infty}\prod_{\mu=1}^{p} dH^\mu dh^\mu dx^\mu P(H^\mu)P(h^\mu)P(x^\mu)\\ \times\,\delta\left(Q-\frac{1}{2p}\sum_{\mu=1}^{p}[1-\tanh^2(H^\mu)][1-\tanh^2(h^\mu)]^2(x^\mu)^2\right)\end{aligned} \qquad (26)$$

leads to

$$\langle Q\rangle = \frac{1}{2}\sigma_x^2\int_{-\infty}^{\infty} dH\,dh\,P(H)P(h)[1-\tanh^2(H)][1-\tanh^2(h)]^2\,, \qquad (27)$$

where both $P(H)$ and $P(h)$ are normal distributions with zero mean and variances σ_H^2 and σ_h^2, respectively.

As before, these integrals are easily evaluated in two limits. For $\sigma_H^2 \gg 1$ and $\sigma_h^2 \gg 1$ we obtain

$$\langle Q\rangle = \frac{2}{3\pi}\frac{\sigma_x^2}{\sigma_H\sigma_h}\,, \qquad (28)$$

while for $\sigma_H^2 \ll 1$ and $\sigma_h^2 \ll 1$ the result is again

$$\langle Q \rangle = \frac{1}{2}\sigma_x^2 \,. \tag{29}$$

B. The Distribution $P(z)$

We now calculate the distribution $P(z)$ for the logarithm $z = \log s$ of the saliencies,

$$P(z) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} dw\, dW \times \delta(z - \log w^2 W^2 \langle Q \rangle) P_w(w) P_W(W) \,. \tag{30}$$

In order to perform the integral over w we note that the zeroes of the argument of the delta function occur at $w = \pm w_o$, with $w_o = \sqrt{\mathrm{e}^z/(W^2\langle Q \rangle)}$. Thus

$$\delta(z - \log w^2 W^2 \langle Q \rangle) = \frac{w_o}{2}[\delta(w + w_o) + \delta(w - w_o)] \tag{31}$$

and

$$\begin{aligned} P(z) &= \frac{1}{2}\int_{-\infty}^{\infty} dW\, w_o P_W(W) \int_{-\infty}^{\infty} dw P_w(w) \\ &\quad \times [\delta(w + w_o) + \delta(w - w_o)] \qquad (32) \\ &= \sqrt{\frac{\mathrm{e}^z}{\langle Q \rangle}} \int_{-\infty}^{\infty} \frac{dW}{|W|} P_W(W) P_w\left(\sqrt{\frac{\mathrm{e}^z}{W^2\langle Q \rangle}}\right), \end{aligned}$$

based on the parity $P_w(w) = P_w(-w)$. Since $P_W(W)$ is also an even function,

$$\begin{aligned} P(z) &= 2\sqrt{\frac{\mathrm{e}^z}{\langle Q \rangle}} \int_0^{\infty} \frac{dW}{W} P_W(W) P_w\left(\sqrt{\frac{\mathrm{e}^z}{W^2\langle Q \rangle}}\right) \\ &= \sqrt{\frac{\mathrm{e}^z}{\langle Q \rangle}} \frac{1}{2\pi\sigma_w\sigma_W} \int_0^{\infty} dW \\ &\quad \times \frac{2}{W} \exp\left[-\frac{1}{2}\left(\frac{\mathrm{e}^z}{\sigma_w^2 W^2 \langle Q \rangle} + \frac{W^2}{\sigma_W^2}\right)\right] . \end{aligned}$$

The change of variables $W^2 = \sqrt{\mathrm{e}^z/\langle Q \rangle}(\sigma_W/\sigma_w)V$ leads to

$$P(z) = \frac{\alpha(z)}{2\pi} \int_0^{\infty} \frac{dV}{V} \exp\left[-\frac{1}{2}\alpha(z)\left(\frac{1}{V} + V\right)\right] , \tag{33}$$

with

$$\alpha(z) = \sqrt{\frac{\mathrm{e}^z}{\langle Q \rangle}} \frac{1}{\sigma_w \sigma_W} , \tag{34}$$

The integral is rewritten in terms of $u = \log V$,

$$P(z) = \frac{\alpha(z)}{2\pi} \int_{-\infty}^{\infty} du \exp[-\alpha(z)\cosh(u)] \tag{35}$$

and identified as a modified Bessel function of the second kind:

$$K_0(x) = \int_0^{\infty} du\, \mathrm{e}^{-x\cosh(u)} \tag{36}$$

to obtain

$$P(z) = \frac{\alpha(z)}{\pi} K_0(\alpha(z)) \,. \tag{37}$$

References

1. D. MacKay 1992, "Bayesian interpolation," *Neural Computation* **4**, 415–447.
2. Y. Chauvin 1988, "A back–propagation algorithm with optimal use of hidden units," in *Advances in Neural Information Processing Systems 1* (Denver, 1988), ed. D. S. Touretzky, pp. 519–526.
3. S. Ramachandran and L. Y. Pratt 1991, "Information measure based skeletonization," in *Advances in Neural Information Processing Systems 4* (Denver, 1991), eds. R. P. Lippmann, J. E. Moody and D. S. Touretzky, pp. 1080–1087.
4. M. C. Mozer and P. Smolensky 1988, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in *Advances in Neural Information Processing Systems 1* (Denver, 1988) ed. D. S. Touretzky, pp. 107–115.
5. Y. Le Cun, J. S. Denker and S. A. Solla 1989, "Optimal brain damage," in *Advances in Neural Information Processing Systems 2* (Denver, 1989), ed. D. S. Touretzky, pp. 598–605.
6. B. Hassibi and D. Stork 1992, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in Neural Information Processing Systems 5* (Denver, 1992), eds. S. J. Hanson, J. D. Cowan and C. L. Giles, pp. 164-171.
7. A. S. Weigend, D. E. Rumelhart and B. A. Huberman 1990, "Generalization by weight–elimination with application to forecasting," in *Advances in Neural Information Processing Systems 3* (Denver, 1990), eds. R. P. Lippmann, J. E. Moody and D. S. Touretzky, pp. 875–882.
8. H. H. Thodberg 1990, "Improving generalization of neural networks through pruning," *IJNS* **1**, 317–326.
9. V. Tresp, R. Neuneier and H. G. Zimmermann 1996, "Early brain damage," in *Advances in Neural Information Processing Systems 9* (Denver, 1996), eds. M. C. Mozer, M. I. Jordan and T. Petsche, pp. 669–675.
10. V. Vapnik 1991, "Principles of risk minimization for learning theory," in *Advances in Neural Information Processing Systems 4* (Denver, 1991), eds. J. E. Moody, S. J. Hanson and R. P. Lippmann, pp. 831–838.

11. J. Gorodkin, L. K. Hansen, A. Krogh, C. Svarer and O. Winther 1993, "A quantitative study of pruning by optimal brain damage," *IJNS* **4**, 159–169.
12. J. Gorodkin, A. Sørensen and O. Winther 1993, "Neural networks and cellular automata complexity," *Complex Systems* **7**, 1–23.
13. G. Cybenko 1989, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signal, and Systems* **2**, 303–314.
14. K. Hornik, M. Stinchcombe and H. White 1989, "Multilayer feedforward networks are universal approximators," *Neural Networks* **2**, 359–366.
15. S. A. Solla, E. Levin and M. Fleisher 1988, "Accelerated learning in layered neural networks," *Complex Systems* **2**, 625–640.
16. J. Gorodkin 1994, "Architectures and complexity of layered neural networks," Cand. Scient. Thesis, CONNECT, The Niels Bohr Institute.
17. T. M. Cover and J. A. Thomas 1991, *Elements of Information Theory* (Wiley, USA), p. 18.
18. J. Denker, D. Schwartz, B. Wittner, S. A. Solla, R. Howard, L. Jackel and J. Hopfield 1987, "Automatic learning, rule extraction, and generalization," *Complex Systems* **1**, 877–922.
19. S. A. Solla 1988, "Learning and generalization in layered neural networks: the contiguity problem," in *Neural Networks: From Models to Applications* (Paris, 1988), eds. L. Personnaz and G. Dreyfus, pp. 168–177.