Why feed-forward networks are in a bad shape

Patrick van der Smagt Gerd Hirzinger
Institute of Robotics and System Dynamics
German Aerospace Center (DLR Oberpfaffenhofen)
82230 Wessling, GERMANY
email smagt@dlr.de

Abstract

It has often been noted that the learning problem in feed-forward neural networks is very badly conditioned. Although, generally, the special form of the transfer function is usually taken to be the cause of this condition, we show that it is caused by the manner in which neurons are connected. By analyzing the expected values of the Hessian in a feed-forward network it is shown that, even in a network where all the learning samples are well chosen and the transfer function is not in its saturated state, the system has a non-optimal condition.

We subsequently propose a change in the feed-forward network structure which alleviates this problem. Benchmarks are used to demonstrate the positive influence of this approach.

1 Introduction

It has long been known [2, 4, 5, 8, 9, 11] that learning in feed-forward networks is a difficult problem, and that this is intrinsic to the structure of such networks. The cause of the learning difficulties is reflected in the Hessian matrix of the learning problem, which consists of second derivatives of the error function. In a recent article, McKeown *et al.* [5] numerically analyze various learning problems and demonstrate that the Hessian is very badly conditioned. This means that the error function has a very strongly elongated shape (a condition κ means that the error function is κ times steeper in its steepest than in its least steep direction); indeed, conditions of 10^{20} are no exception in feed-forward network learning, and in fact mean that the problem exceeds the representational accuracy of the computer.

We will show that this problem is caused by the structure of the feed-forward network. An adaptation to the learning rule, which is reflected by an extra connection in the network, is shown to improve this condition.

2 The learning problem

We define a feed-forward neural network with a single layer of hidden units as

$$\mathcal{N}(\vec{x}, W)_o = \sum_h w_{ho} s \left(\sum_i w_{ih} x_i + \theta_h \right) \tag{1}$$

where i indicates the i'th input, h the h'th hidden unit, o the o'th output, and $\vec{x}$ is an input vector $(x_1, x_2, \ldots, x_{N_i})$. The total number of weights w_{ij} is n. W.l.o.g. we will assume $o = 1$ in the sequel. The learning task consists of minimizing an approximation error

$$E_{\mathcal{N}}(W) = \frac{1}{2} \sum_{p=1}^{\Pi} \left\| \mathcal{N}(\vec{x}^{(p)}, W) - \vec{y}^{(p)} \right\| \tag{2}$$

[0]To be presented at the ICANN 1998.

where Π is the number of learning samples and $\vec{y}$ is an output vector $(y_1, y_2, \dots, y_{N_o})$. $\|\cdot\|$ is usually taken to be the L_2 norm. The error E can be written as a Taylor expansion around W_0:

$$E_{\mathcal{N}}(W + W_0) = E(W_0) - J^T W + \frac{1}{2} W^T H W + r(W_0). \tag{3}$$

Here, H is the Hessian and J is the Jacobian of E around W_0. In the case that E is purely quadratic, the rest term is 0 and a Newton-based second-order search technique can then be used to find the extremum in the ellipsoid E. A problem arises, however, when the axes of this ellipsoid are very different. When the ratio of the lengths of the largest and smallest axes is very large (close to the computational precision of the computer used in optimization), the computation of the exact local gradient will be imprecise, such that the system is difficult to minimize. Now, since H is a real symmetric matrix, its eigenvectors span an orthogonal basis, and the directions of the axes of E are equal to the eigenvectors of H. Furthermore, the corresponding eigenvalues are the square root of the lengths of the axes. Therefore, the condition number κ of the Hessian matrix, which is defined as the ratio of the largest and smallest singular values (and therefore, for a positive definite matrix, of the largest and smallest eigenvalues), determines how well the error surface E can be minimized.

3 The linear network

In the case that $s(\cdot)$ is the identity, $\mathcal{N}(W)$ is equivalent to a linear feed-forward network $\mathcal{N}(W')$ without hidden units. Then the rest term of Eq. (3) is zero and we can write

$$E_{\mathcal{N}}(W' + W_0') = E(W_0') - J^T W' + \frac{1}{2} W'^T H W'. \tag{4}$$

In the linear case the Hessian reduces to (leaving the index (p) out):

$$H_{jk} = \Pi^{-1} \sum_{p=1}^{\Pi} x_j x_k, \qquad 1 \le j, k \le n = N_i + 1 \tag{5}$$

where, for notational simplicity, we set $x_{N_i+1} \equiv 1$. In this case H is the covariance matrix of the input patterns. It is instantly clear that H is a positive definite symmetric matrix.

Le Cun *et al.* [3, 4] show that, when the input patterns are uncorrelated, H has a continuous spectrum of eigenvalues $\lambda_- < \lambda < \lambda_+$, where $\lambda_\pm = \frac{n}{\Pi}(1 \pm \sqrt{\Pi/n})^2 \nu(x_k)$. Furthermore, there is one eigenvalue λ_n of multiplicity order n present only in the case that $\langle x_k \rangle \ne 0$. Therefore, the Hessian for a linear feed-forward network is optimally conditioned when $\langle x_k \rangle = 0$.

The reason for this behaviour is very understandable. As $\Pi \to \infty$, the summation of uncorrelated elements $x_i x_j$ will cancel out when $\langle x \rangle = 0$, except where $i = j$, i.e., on the diagonal of the covariance matrix. In the limit these diagonal elements go towards the variance of the input data $\nu(x_i) = \sum_p x_i^{(p)2}$. Gerschgorin's theorem shows that every eigenvalue of a matrix A lies in at least one circular disk with center a_{ii} and radius $\sum_{i \ne j} |a_{ij}|$. Differently put, the eigenvalues of a diagonal matrix equal the elements on the diagonal.

From this analysis it can be concluded that centering of the input signals greatly improves the condition of the learning system, and an optimization technique will be numerically stable [1].

4 Multi-layer feed-forward networks

In the case that a nonlinear feed-forward network is used, i.e., $s(\cdot)$ is a nonlinear transfer function, the rest term in Eq. (3) cannot be neglected in general. However, it is a well-known fact [6, 7] that the rest term $r(\cdot)$ is negligible 'close enough to a minimum.' From the definition of $E_{\mathcal{N}}$ we can compute that

$$H_{j,k} = \frac{1}{2\Pi}\sum_{p=1}^{\Pi}\frac{\partial^2\left[\mathcal{N}(\vec{x})-y\right]^2}{\partial w_j \partial w_k} \tag{6}$$

$$= \Pi^{-1}\sum_{p=1}^{\Pi}\left[\mathcal{N}(\vec{x})-y\right]\frac{\partial^2\mathcal{N}(\vec{x})}{\partial w_j \partial w_k} + \frac{\partial\mathcal{N}(\vec{x})}{\partial w_j}\frac{\partial\mathcal{N}(\vec{x})}{\partial w_k}. \tag{7}$$

We investigate the properties of $H_{j,k}$ of Eq. (7). The first term of the Hessian has a factor $[\mathcal{N}(\vec{x})-y]$. Close to a minimum, this factor is close to zero such that it can be neglected. Also, when summed over many learning samples, this factor equals the random measurement error and cancels out in the summation. Therefore we can write

$$H_{j,k} \approx \Pi^{-1}\sum_{p=1}^{\Pi}\frac{\partial\mathcal{N}(\vec{x})}{\partial w_j}\frac{\partial\mathcal{N}(\vec{x})}{\partial w_k}. \tag{8}$$

5 Properties of the Hessian

Every Hessian of a feed-forward network with one layer of hidden units can be partitioned into four parts, depending on whether the derivative is taken with respect to a weight from input to hidden or from hidden to output unit. We take the simplification of Eq. (8) as a starting point. The partial derivative of $\mathcal{N}$ can be computed to be

$$\frac{\partial\mathcal{N}(\vec{x})}{\partial w_{ho}} = s\left(\sum_i w_{ih}x_i + \theta_h\right) \equiv a_h, \qquad \frac{\partial\mathcal{N}(\vec{x})}{\partial w_{ih}} = x_i w_{ho} s'(a_h) \equiv x_i w_{ho} a'_h \tag{9}$$

where $s'(\cdot)$ is the derivative of $s(\cdot)$. Using (9) we can write the Hessian as a block matrix

$$\Pi H \approx \left[\begin{array}{c|c} \sum_p (x_{i_1} w_{h_1 o} a'_{h_1})(x_{i_2} w_{h_2 o} a'_{h_2}) \equiv {}^{00}H & \sum_p (x_i w_{h_2 o} a'_{h_2}) a_{h_1} \equiv {}^{01}H \\ \hline \sum_p (x_i w_{h_1 o} a'_{h_1}) a_{h_2} \equiv {}^{10}H & \sum_p a_{h_1} a_{h_2} \equiv {}^{11}H \end{array}\right] \tag{10}$$

(note that ${}^{10}H^T = {}^{01}H$). Now, minimal requirements for this matrix to be well-conditioned are that (a) each of its submatrices is well-conditioned, (b) the elements of all submatrices are of the same order, and (c) the elements of the matrices are uncorrelated.

Assuming that the input samples have a normal (0,1) distribution, we can compute the expectations and variances of the elements in H by determining the distribution functions of these elements. It can be shown that these distribution functions are

$$F_{{}^{00}H_{i=j}}(x) = \frac{1}{\sqrt{\Pi}}\frac{1}{2\sqrt{x}}\frac{\exp\left[-\mathrm{atanh}^2\left(\sqrt{1-\sqrt{x}}\right)/(2n)\right]}{\sqrt{2n\pi}2x\sqrt{1-\sqrt{x}}}, \tag{11}$$

$$F_{{}^{00}H_{i\neq j}}(x) = \frac{1}{\sqrt{\Pi}}\frac{\exp\left[-\mathrm{atanh}^2\left(\sqrt{1-x}\right)/(2n)\right]}{\sqrt{2n\pi}2x\sqrt{1-x}}, \tag{12}$$

$$F_{{}^{11}H_{i=j}}(x) = \frac{1}{\sqrt{\Pi}}\frac{1}{2\sqrt{x}}\frac{\exp\left[-\mathrm{atanh}^2(\sqrt{x})/(2n)\right]}{\sqrt{2n\pi}(1-x)}, \tag{13}$$

$$F_{{}^{11}H_{i\neq j}}(x) = \frac{1}{\sqrt{\Pi}}\frac{\exp\left[-\mathrm{atanh}^2(x)/(2n)\right]}{\sqrt{2n\pi}(1-x^2)}. \tag{14}$$

These equations describe the elements of the Hessian in the case that the weights and input samples are all optimally chosen, and therefore the hidden neurons are not saturated (i.e., the input to the hidden neurons has a normal(0,1) distribution).

Figure 1 depicts the expectations and standard deviations of the elements of H for $\Pi = 100$. From the figure it can be seen that, even though the network is in an optimal state, the elements of ${}^{11}H$ are much larger than those of ${}^{00}H$. Naturally, this effect is much stronger when weights from input to hidden units are large, such that the hidden units are close to their saturated state. In that case, $a'_h \approx 0$ and the elements of ${}^{00}H$ (as well as those of ${}^{01}H$) tend to 0.

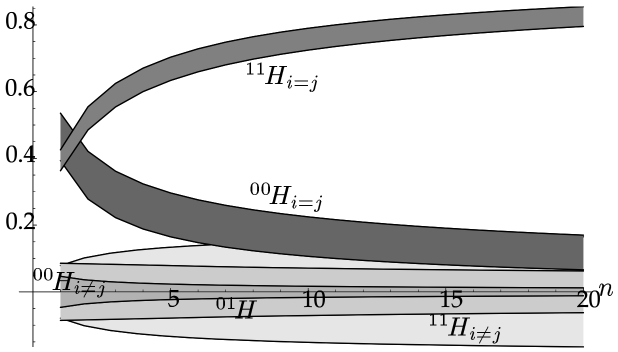

Figure 1: The distribution of the elements of H for $\Pi = 100$.

Centering. The centering method as proposed in [3], when also applied to the activation values of the hidden units, ensures an optimal condition for ^{11}H. Schraudolph and Sejnowski [9] have shown that centering in backpropagation further improves the learning problem. Using argumentation similar to Le Cun *et al.* [3], they furthermore suggest that centering the $\delta_o = y_o - a_o$ as well as the $\delta_h = \sum_o \delta_o w_{ho} s'(a_h)$ improves the condition of H. Although this will improve the condition of the ^{00}H and ^{01}H, the problem that the elements of ^{00}H and ^{11}H are very different in size remains. We suggest that this approach alone is not sufficient to improve the learning problem.

6 An adapted learning rule

To understand *why* the elements of H are so different, we have to consider the back-propagation learning rule. First, Saarinen *et al.* [8] list a few cases in which the Hessian may become (nearly) singular. The listed reasons are associated with the ill character of the transfer functions which are customarily used: the sigmoidal function $s(x)$, which saturates (i.e., the derivative becomes zero) for large input. However, another problem exists when a network has a small weight leaving from a hidden unit, the influence of the *weights that feed into this hidden unit* is significantly reduced. This problem touches a characteristic problem in feed-forward network learning: the gradients in the lower-layer weights are influenced by the higher-layer weights. Why this is so can be seen from the back-propagation learning method, which works as follows. For each learning sample:

1. compute $\delta_o = y_o - a_o$ where a_o is the activation value for output unit o;
2. compute $\Delta w_{ho} = \delta_o a_h$ where a_h is the activation value for hidden unit h;
3. compute $\delta_h = \sum_o \delta_o w_{ho} s'(a_h)$;
4. compute $\Delta w_{ih} = \delta_h x_i = \sum_o \delta_o w_{ho} s'(a_h) x_i$.

The gradient is then computed as the summation of the Δw's. The gradient for a weight from an input to a hidden unit becomes negligible when δ_o is small (i.e., the network correctly represents this sample), x_i is small (i.e., the network input is close to 0), w_{ho} is small or $s'(a_h)$ is small (because w_{ih} is large). The latter two of these cases are undesirable, and lead to paralysis of the weights from input to hidden units.

In order to alleviate these problems, we propose a change to the learning rule as follows:

$$\Delta w_{ih} = \sum_o \delta_o (w_{ho} s'(a_h) + a_h) x_i = \delta_h x_i + a_h x_i \sum_o \delta_o. \tag{15}$$

By adding a_h to the middle term, we can solve both paralysis problems. In effect, an extra connection from each input unit to each output unit is created, with a weight value coupled to the

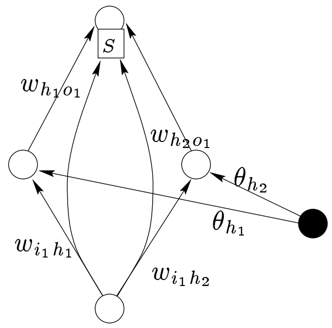

Figure 2: An exemplar linearly augmented feed-forward neural network.

weight from the input to hidden unit. The o'th output of the neural network is now computed as

$$\mathcal{M}(\vec{x}, W)_o = \sum_h \left(w_{ho} s \left[\sum_i w_{ih} x_i + \theta_h \right] + S \left[\sum_i w_{ih} x_i \right] \right) \tag{16}$$

where $dS(x)/dx \equiv s(x)$. In the case that $s(x)$ is the tanh function we find that $S(x) = \log\cosh x$; note that this function asymptotically goes to $|x| - \log 2$. In effect, *we add the absolute of the hidden unit activation values to each output unit*. We call the new network the **linearly augmented feed-forward network**. The structure of this network is depicted in figure 2 for one input and output unit and two hidden units.

6.1 Analysis of the Hessian's condition

We can compute the approximation error E' for $\mathcal{M}$ similar to (2) and construct the Hessian H' for $\mathcal{M}$. We can relate the quadrants of H' to those of H as follows. First, ${}^{11}H' = {}^{11}H$, and

$$\begin{aligned} {}^{00}H'_{i_1+h_1N_i;i_2+h_2N_i} &= {}^{00}H_{i_1+h_1N_i;i_2+h_2N_i} + \Pi^{-1}\sum_p x_{i_1}a_{h_1}x_{i_2}a_{h_2}(1 + w_{h_1o}a'_{h_1} + w_{h_2o}a'_{h_2}), \\ {}^{01}H'_{i+h_1N_i;i+h_2N_i} &= {}^{01}H_{i+h_1N_i;i+h_2N_i} + \Pi^{-1}\sum_p x_i a_{h_1} a_{h_2}. \end{aligned}$$

Using the same line of argument as in section 3 we note that it is important that the a_i's be centered, i.e., (a) the input values should be centered, and (b) it is advantageous to use a centered hidden unit activation function (e.g., $\tanh(x)$ rather than $1/(1+e^{-x})$).

6.2 $\mathcal{M}$ and the Universal Approximation Theorems

It has been shown in various publications that the ordinary feed-forward neural network $\mathcal{N}$ can represent any Borel-measurable function with a single layer of hidden units which have sigmoidal or Gaussian activation functions. It can be easily shown [11] that $\mathcal{N}$ and $\mathcal{M}$ are equivalent, such that all representation theorems that hold for $\mathcal{N}$ also hold for $\mathcal{M}$.

7 Examples

The new method has been tested on a few problems. First, XOR classification with two hidden units. Secondly, the approximation of $\sin(x)$ with 3 hidden units and 11 learning samples randomly chosen between 0 and 2π. Third, the approximation of the inverse kinematics plus inverse perspective transform [12] for a 3 DoF robot arm with camera fixed in the end-effector. The network consisted of 5 inputs, 8 hidden units, and 3 outputs; the 1107 learning samples were gathered using a Manutec R3 robot. All networks were trained with Polak-Ribière conjugate gradient with Powell restarts [10].

All experiments were run 1000 times with different initial weights. The results are illustrated below. Note that, for the chosen problems, $\mathcal{M}$ effectively smoothes away local minima and saddle points (% stuck goes to 0.0). The reported E was measured only over those runs which did not get stuck.

		$\mathcal{N}$	$\mathcal{M}$
XOR	% stuck	22.4	0.0
	# steps to reach $E = 0.0$	189.1	65.3
sin	% stuck	42.3	0.0
	E after 1000 iterations	$29 \cdot 10^{-5}$	$7.3 \cdot 10^{-5}$
robot	E after 1000 iterations	$5.3 \cdot 10^{-3}$	$1.8 \cdot 10^{-3}$

References

[1] Å. Björk, E. Grimme, and P. van Dooren. An implicit shift bidiagonalization algorithm for ill-posed problems. *BIT*, 34:510–534, 1994.

[2] A. Krogh and J. A. Hertz. Dynamics of generalization in linear perceptrons. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*, pages 897–903, San Mateo, CA, 1991. Morgan Kaufmann Publishers.

[3] Y. Le Cun, I. Kanter, and S. A. Solla. Eigenvalues of covariance matrices: Application to neural network learning. *Physical Review Letters*, 66(18):2396–2399, 1991.

[4] Y. Le Cun, I. Kanter, and S. A. Solla. Second order properties of error surfaces: Learning time and generalization. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, pages 918–924, San Mateo, California, 1991. Morgan Kaufman.

[5] J. J. McKeown, F. Stella, and G. Hall. Some numerical aspects of the training problem for feed-forward neural nets. *Neural Networks*, 10(8):1455–1463, 1997.

[6] E. Polak. *Computational Methods in Optimization*. Academic Press, New York, 1971.

[7] M. J. D. Powell. Restart procedures for the conjugate gradient method. *Mathematical Programming*, 12:241–254, April 1977.

[8] S. Saarinen, R. Bramley, and G. Cybenko. Ill-conditioning in neural network training problems. *Siam Journal of Scientific Computing*, 14(3):693–714, May 1993.

[9] N. N. Schraudolph and T. J. Sejnowski. Tempering backpropagation networks: Not all weights are created equal. In D. S. Touretzky, M. C. Moser, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, pages 563–569, 1996.

[10] P. van der Smagt. Minimisation methods for training feed-forward networks. *Neural Networks*, 7(1):1–11, 1994.

[11] P. van der Smagt and G. Hirzinger. Solving the ill-conditioning in neural network learning. In J. Orr, K. Müller, and R. Caruana, editors, *Tricks of the Trade: How to Make Neural Networks Really Work*. Lecture Notes in Computer Science, Springer Verlag, 1998. In print.

[12] Patrick van der Smagt. *Visual Robot Arm Guidance using Neural Networks*. PhD thesis, Dept of Computer Systems, University of Amsterdam, March 1995.