A QUANTITATIVE STUDY OF PRUNING BY OPTIMAL BRAIN DAMAGE

J. Gorodkin[a)], L.K. Hansen[b)], A. Krogh[b)], C. Svarer[b)], and O. Winther[a)]

[a)]CONNECT, Niels Bohr Institute
Blegdamsvej 17, DK-2100 København Ø, Denmark

[b)]CONNECT, Electronics Institute
Technical University of Denmark, B. 349, DK-2800 Lyngby, Denmark

SUPERrevised, June 1993

Abstract

The Optimal Brain Damage (OBD) scheme of Le Cun, Denker, and Solla, for pruning of feed-forward networks, has been implemented and applied to the contiguity classification problem. It is shown that OBD improves the learning curve (the test error as function of the number of examples). By inspecting the architectures obtained through pruning it is found that the networks with less parameters have the smallest test error, in agreement with "Ockhams Razor". Based on this, we propose a heuristic which selects the smallest successful architecture among a group of pruned networks and we show that it leads to very efficient optimization of the architecture. The validity of the approximations involved in OBD are discussed and it is found and they are surprisingly accurate for the problem studied.

1 Introduction

Much attention is currently directed towards the problem of identifying optimal *application-specific* neural network architectures. It is generally believed that good generalization is associated with minimal representation and it may indeed be proven within many models that the generalization (or test) error is decreased by removing superfluous resources.

Pruning schemes have been successfully applied to layered neural networks [11, 12, 16, 17, 7]. Apart from the identification of more compact representations, significant improvements in performance have been obtained. The *Optimal Brain Damage* (OBD) scheme of Le Cun, Denker and Solla [12], stands out for its success in reducing the complexity of a well-trained network for identification of hand-written ZIP-codes [11, 12]. The method is based on the computation of the *saliency* of a weight, which is an estimate of the increase in training error induced by deleting the given weight.

Recently Hassibi and Stork proposed a generalization of OBD, the Optimal Brain Surgeon (OBS) [7, 8]. OBS was first found to perform well on a set of Boolean problems [7], and it has later been applied to the NETtalk problem. While OBD is based on the diagonal approximation of the second derivative matrix (the *Hessian*), OBS uses the full Hessian, which makes it possible to estimate the combined effect of pruning and retraining — assuming the second order approximation is valid, retraining is not necessary in OBS. Because of the approximations entering OBD, it is recommended to use it iteratively: alternating pruning of a small set of weights followed by retraining [11]. When applied to larger problems like NETtalk, Hassibi *et al.* adapted the OBS prescription to also include retraining, and reduced the complexity of the full Hessian by a block-diagonal approximation [8].

In this study we provide some quantitative results on the performance of OBD for Boolean classification problems. A separate presentation addresses regression problems[15]. In the original studies [11, 12] OBD was used to prune an architecture already optimized by hand (the zip-code reading network). One of the main motivations for this work is to see if OBD can prune a large redundant and unstructured network to a near-optimal one with only a fraction of the original number of weights.

For the *contiguity* classification problem we show numerically that OBD indeed enhances the learning performance, and in particular that the learning curve (*i.e.*, the generalization ability as a function of the size of the training set) is improved by OBD. The level of generalization is higher and the *learning transition*, a "knee" in the learning curve which identifies the minimum number of training examples needed for good generalization [13], occur earlier for the networks pruned by OBD. A fully connected network can thus be used if sufficient training examples are available, but there is a range of training set sizes for which OBD is crucial for successful learning. The scaling of this improvement with problem size is an open problem, *i.e.*, whether learnability in the sense of Valiant [19] is affected by OBD.

In this work we show that OBD may be further improved by a strategy in which a group of networks are pruned and the smallest final network picked, leading to a dramatic improvement in generalization ability. The number of training examples needed to perfectly learn our particular test problem consistently drops to around 100, whereas at least 200 examples are needed to obtain a fully connected network that generalizes well. Even a majority vote among 10 fully connected independently trained networks has a learning transition around 150 examples. This shows that OBD is capable of finding a near-optimal network architecture in a moderate number of trails. When the smallest of 10 networks is chosen the learning transition becomes very sharp: with 90 examples the generalization error is around 40%, and above 100 examples it is very close to zero.

In order to check the validity of the approximations used in OBD, we compare the estimated saliencies with the observed ones, finding surprising agreement. This is in line with the results of Le Cun *et al.* on recognition of handwritten digits [12].

The paper is organized as follows: in the next section OBD is derived and some of it's inherent difficulties are discussed. Our specific two-layer feed forward network is introduced in section three, and the saliency estimators are derived and discussed. Section four contains the results of our numerical investigations of OBD. The validity of the various approximations entering the saliency estimation is discussed, and a result for the "monks1 problem" is presented.

2 Pruning by Optimal Brain Damage

Present generalization theories [1, 5, 9, 14] estimate the test error from the training error and model capacity. The basic idea is that the superfluous parameters allow for too many generalizations from the training set, hence the "correct" generalization is picked with a small probability. The design rule derived from this insight would be:

> Minimize the capacity of the network under the constraint that it is able to implement the training set.

This strategy is often referred to as *Ockhams Razor*, see *e.g.* [16], since the above principle is a special form of a general guideline proposed by the medieval philosopher. A difficulty appears when applying the razor to problems with noise or inconsistencies in the training set, because a certain level of the training error is unavoidable, and in fact zero training error would lead to poor generalization ability. See [15] for a discussion of this problem in the context of non-linear regression. Throughout this presentation we restrict ourselves to the case of a noise free teacher that provides only correct examples.

Assuming that the network capacity is an increasing function of the number of parameters [3], Ockhams Razor may be implemented by *brute force* pruning as proposed by Thodberg [16, 17]. Alternatively, following Le Cun *et al.* [12], we may attempt to estimate the saliency and use this measure for ranking the parameters as candidates for pruning. For completeness we rederive and discuss the OBD method.

2.1 Weight saliency

Optimal Brain Damage is based on a second order approximation to the increase in training error resulting from elimination of a weight. If the weights w_i are perturbed by δw_i the change in the training error E_T is, to second order,

$$\delta E_T \simeq \sum_{i=1}^{N} \frac{\partial E_T}{\partial w_i} \delta w_i + \frac{1}{2} \sum_{i=1}^{N} \frac{\partial^2 E_T}{\partial w_i^2} \delta w_i^2 + \frac{1}{2} \sum_{j=1}^{N} \sum_{i=1, i \neq j}^{N} \frac{\partial^2 E_T}{\partial w_i \partial w_j} \delta w_i \delta w_j, \tag{1}$$

where N is the number of weights in the network. The first-order term is zero if the training has reached a minimum of E_T. If deletion of only one weight is considered ($\delta w_i = 0$ for all i except $\delta w_j = -w_j$), the off-diagonal part of the second order term is also zero, and all is left is

$$\delta E_T \sim s_j \equiv \frac{1}{2} \frac{\partial^2 E_T}{\partial w_j^2} w_j^2, \tag{2}$$

which is called the *saliency*. Figure 1 illustrates Equation (2). In the original work of Le Cun *et al.* the saliency estimator was used for deleting sets of weights $\{w_k | k \in D\}$, and the cumulative increase was approximated by

$$\delta E_T \sim \sum_{j \in D} s_j, \tag{3}$$

with the additional assumption that off-diagonal terms in the second derivative matrix can be neglected.

If the error function is additive, *i.e.*, $E_T = \frac{1}{p} \sum_{\alpha=1}^{p} E^\alpha$, where E^α is the error on example α, the diagonal elements of the second derivative matrix are given by:

$$\frac{\partial^2 E_T}{\partial w_j^2} = \frac{1}{p} \sum_{\alpha=1}^{p} \left[\frac{\partial^2 E^\alpha}{\partial (V^\alpha)^2} \left(\frac{\partial F_w(x^\alpha)}{\partial w_j} \right)^2 + \frac{\partial E^\alpha}{\partial V^\alpha} \frac{\partial^2 F_w(x^\alpha)}{\partial w_j^2} \right] \tag{4}$$

where F_w is the function implemented by the network, and $V^\alpha = F_w(x^\alpha)$ is the output corresponding to input x^α. The parameter p denotes the number of examples in the training set. For the most commonly used cost functions the second term can be neglected since $\frac{\partial E}{\partial V}$ is proportional to the error on example α (see below).

For the sum of squared errors $E_T = \frac{1}{2p} \sum_\alpha (y^\alpha - F_w(x^\alpha))^2$, Equation (4) becomes

$$\frac{\partial^2 E_T}{\partial w_j^2} = \frac{1}{p} \sum_{\alpha=1}^{p} \left[\left(\frac{\partial F_w(x^\alpha)}{\partial w_j} \right)^2 + (y^\alpha - F_w(x^\alpha)) \left(- \frac{\partial^2 F_w(x^\alpha)}{\partial w_j^2} \right) \right] \tag{5}$$

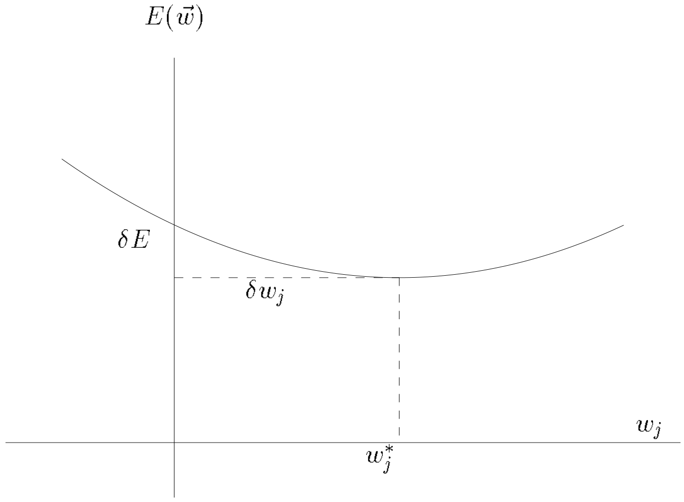

Figure 1: After training the network is assumed to be in a training error minimum (or close to it) with $w_j = w_j^*$ as illustrated. This weight is removed by setting $w_j = 0$. The cost is δE, which is in turn estimated by the saliency.

where the training set consists of the p pairs of input and target (x^α, y^α). The Levenberg-Marquardt approximation is invoked by Le Cun *et al.* [12], to drop the second term. If the network has learned to reproduce the examples exactly, the errors: $e^\alpha = y^\alpha - F_w(x^\alpha)$ would be zero. In general, one would expect that a well-trained network should have "random" errors, *i.e.*, the e^α would have zero mean. The correlation between errors and the second derivative of the network-function is thus expected to vanish for large training sets. An additional motivation for neglecting the second term of Equation (5) is *computational*: the remaining first term of the second derivative matrix is non-negative, and contains quantities already computed during training. The final estimate of the training error increase when deleting weight j is

$$s_j = w_j^2 \frac{1}{2p} \sum_\alpha \frac{\partial^2 E^\alpha}{\partial (V^\alpha)^2} \left(\frac{\partial F_w(x^\alpha)}{\partial w_j} \right)^2 . \tag{6}$$

If E_T is the sum of squared errors, $\frac{\partial^2 E}{\partial V^2} = 1$.

2.2 Inherent problems in OBD

A network pruning method based on the saliencies rests on two crude assumptions. First, the training error is assumed to be well approximated by a second order expansion around its minimum point, even far from it. It is not difficult to think of situations where this is not the case. Assume, for instance, that a standard sigmoid like tanh is used as activation function for a hidden unit which saturates for most inputs. The error surface will be very flat, and the second order expansion will be a poor approximation. A small weight decay regularizer can be used to avoid saturation of the sigmoids; we have found that such a technique greatly improves the performance of OBD.

The second crude assumption is that a large increase in error *before* retraining ($\delta E_{before}(w_j)$) when removing weight w_j also means a large error increase *after* retraining ($\delta E_{after}(w_j)$). Again one can easily think of cases where the ranking of weights according to δE_{before} would differ from the ranking based on δE_{after}. Ideally one would like to use the latter, and that is essentially what the methods of Thodberg [16] and OBS [8] aim at. It is worth noting that since the error always decreases during retraining, δE_{before} is an *upper bound* for δE_{after}, *i.e.*,

$$\delta E_{after}(w_j) \leq \delta E_{before}(w_j). \tag{7}$$

Therefore using OBD might be a good trade-off when considering the computational cost of the two other algorithms mentioned above.

3 OBD for a two-layer network

To learn Boolean classifications of M-dimensional binary inputs, we have chosen a family of functions that map the M-dimensional hypercube into the reals:

$$F_w(x^\alpha) = \sum_{j=1}^{n_H} W_j \tanh\left(\sum_{k=0}^{n_I} w_{jk} x_k^\alpha\right) + W_0, \tag{8}$$

n_H, n_I are the numbers of hidden and input units respectively, (here $n_I = M$). The thresholds are implemented as weights connecting to clamped units, $x_0 = 1$, in the preceding layer. The binary classification is given by the sign of the output. This is a standard two-layer network with one linear output unit, where w_{jk} denotes weights from input to hidden units and W_j the weights from hidden to output. The fully connected architecture is shown in Figure 2.

For training error we use the usual sum of squares:

$$E_T = \frac{1}{2p} \sum_{\alpha=1}^{p} \left(y^\alpha - F_w(x^\alpha)\right)^2. \tag{9}$$

One could argue that a linear output unit is not the obvious choice for a Boolean classification problem. The main virtue of (8) is that it allows for efficient optimization, as shown below.

Saliencies of two different functional forms are obtained, corresponding to the two layers of weights. The saliency of an output weight is:

$$s_j = W_j^2 \frac{1}{2p} \sum_{\alpha=1}^{p} \tanh^2\left(\sum_{k=0}^{n_I} w_{jk} x_k^\alpha\right) \tag{10}$$

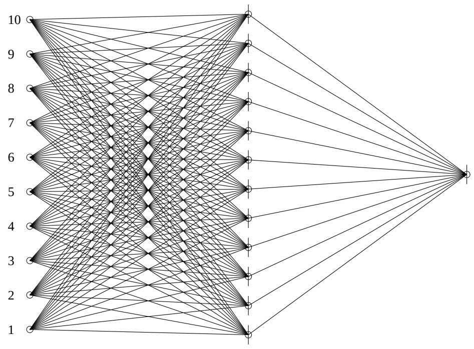

Figure 2: Fully connected architecture (10-12-1), an active threshold is indicated by a vertical line through the unit.

The saliency of an input-to-hidden weight is found to be:

$$s_{jk} = W_j^2 w_{jk}^2 \frac{1}{2p} \sum_{\alpha=1}^{p} \left[1 - \tanh^2 \left(\sum_{k'=0}^{n_I} w_{jk'} x_{k'}^{\alpha} \right) \right]^2 , \qquad (11)$$

since $(x_k^\alpha)^2 = 1$ for Boolean inputs. Note that the training set output (*i.e.*, the specific Boolean function) only enters the saliencies implicitly through the weight distribution of the trained network.

Magnitude based pruning, in which the weights are ranked solely according to magnitude, has been criticized in [12] and [7]. Such criticism can be supported by analysis of equations (11) and (10). In the OBD context, magnitude based pruning is equivalent to the assumption that the second derivative is a constant for all weights. For the hidden to output weights, the second derivative is determined by the activation level of the hidden unit, which need not be the same for all hidden units. The second derivative with respect to an input-to-hidden weight includes the corresponding output weight, which need not be the same for all hidden units either.

Note that the accumulated saliency of a deleted set of weights is an estimate of the increase in the *training error*. If we adopt the standard procedure and let the network classification be determined by the *sign* of the output, some problems might tolerate significant increases in squared error, without making training set classification errors. This observation suggests a *stop criterion*: in the "noise-free" case, where all examples are correct, we prune the networks until classification errors appear on the training set.

4 Experiments

4.1 The contiguity problem

The contiguity (or "two or three clumps") problem was discussed in Denker *et al.* [5], and used by Thodberg for testing a pruning algorithm [16]. The Boolean input field (± 1) is interpreted as a one dimensional "image", and connected clumps of $+1's$ are counted. Two classes of patterns are considered: those with two and three clumps. The Boolean function is defined by assigning the output -1 to the two-clump inputs and $+1$ to the three-clump inputs. In particular we study the $n_I = 10$ case which has a total of 792 legal input patterns of which 432 have three clumps and 360 have two clumps. It is known that this problem can not be solved by a simple perceptron, *i.e.*, it is not linearly separable and hidden units are needed. A simple symmetric solution is illustrated in Figure 3; only a few (shared) parameters are necessary. In this network the hidden units work as *edge-detectors*. It was shown that only 20 to 30 examples are needed to obtain good generalization if this architecture is chosen [13].

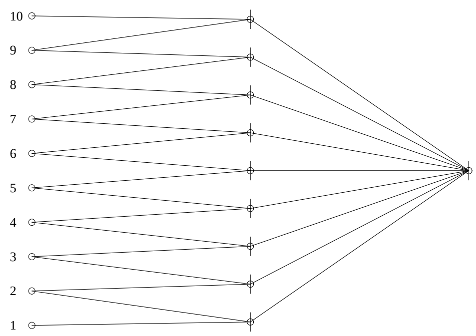

Figure 3: A sparse symmetric architecture for the contiguity problem

4.2 The simulator

The learning algorithm is based on mixed back-propagation and second order batch mode optimization. Since we have chosen a linear output unit, the cost function is quadratic in the hidden-to-output weights, allowing for direct optimization of these weights by matrix inversion [2]. The weights to the hidden units are initially adapted by batch mode gradient descent, and then refined by the second order *pseudo-Newton Rule* [9]:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}} \Big/ \frac{\partial^2 E}{\partial {w_{ij}}^2} \,. \tag{12}$$

The parameter η is used to secure that all weight updates in the "second order phase" lead to a decrease in the cost function. The learning rate η is initialized to 1 before each step, and iteratively diminished by powers of two until the step leads to a decrease in the cost function. We proceed by iteration, alternating updates between the two layers of weights.

To ensure numerical stability and to assist in the pruning procedure we add to the cost function a small *weight decay* term [9]:

$$E = E_T + E_W = \frac{1}{2p} \sum_{\alpha=1}^{p} (y^\alpha - F_w(x^\alpha))^2 + \frac{\epsilon}{2p} \left(\sum_{j=1}^{n_H} \sum_{k=0}^{n_I} w_{jk}^2 + \sum_{j=0}^{n_H} W_j^2 \right), \tag{13}$$

with $\epsilon = 0.005$. Calculating the derivatives, the update rule for the weights in the first layer of the network is given by:

$$\Delta w_{jk}(t) = -\eta \left(\frac{\partial E_T}{\partial w_{jk}} + \frac{\epsilon}{p} w_{jk} \right) \Big/ \left(\frac{\partial^2 E_T}{\partial {w_{jk}}^2} + \frac{\epsilon}{p} \right). \tag{14}$$

The matrix inversion scheme is also regularized by the weight decay term.

In each iteration of the pruning session, learning proceeds until no further decrease of the cost function can be obtained. The saliencies are then estimated and used to rank the weights. In our present version of OBD we proceed as in [12], and speed up the pruning procedure by removing several weights in each iteration (5% of the remaining weights). After pruning, the network is retrained using the mixed second order scheme described above. The procedure is carried out iteratively and stops when the network cannot be retrained to zero classification errors on the training set. This stop criterion is meaningful as long as the training set contains no false examples.

4.3 Learning curves

Our main results are shown in figures 4-5: learning curves for a fully connected network and networks trained and pruned using OBD. For each training set size, ten experiments were performed with a randomly chosen (but fixed) training set and random initial configurations of the weights. No strong dependency was observed on the particular choice of training set.

These learning curves comply with statistical theories on generalization: an S-shaped learning curve with a learning transition characteristic of the given problem and the given architecture. The location of this transition is a useful measure of the suitability of an architecture or an architecture optimization algorithm, to a specific task. Figure 4 shows the learning curve for the fully connected network. The learning transition is located around 200 examples: note that even with more than 200 examples some networks perform almost as bad as random guessing. The characteristics of the learning curve do not depend on the particular training sets chosen; the fluctuation observed is due to the success or failure of the learning algorithm (keep in mind that each network is error free on the given training set). The scatter in the performance of the solutions could be stabilized by a majority vote of the *ensemble* [6], as indicated by the line in the figure.

Figure 5 shows the learning curve for the pruned networks, where the learning transition is now located at only 150 examples. The fluctuations are significant, so the transition is not a sharp one. Picking the smallest network among the solutions obtained results in the line shown in the figure. Although based on simple heuristics, the strategy is remarkably stable and it produces a very sharp learning transition just below 100 examples. Figure 6 illustrates the correlation between network size and test error for a set of 50 networks trained on 140 examples, and it provides strong evidence in favor of Ockhams Razor, as found by Thodberg [16]. We conclude that OBD can be improved by a ensemble search combined with Ockhams Razor.

An example of the optimized architectures is shown in Figure 7. This particular architecture was trained and pruned on 160 examples. It makes no errors on the legal inputs. It is interesting to compare this network with the symmetric one Figure 3. The solution obtained through pruning uses only eight hidden units and 36 connections (including thresholds), while the symmetric network has nine hidden units and 37 connections. While the symmetric network is easy to interpret, it is not at all obvious how the network of Figure 7 works. Some of the hidden units are apparently "edge-detectors", while others are more complex (triple input) feature detectors, but these features are still local.

4.4 Test of the saliency estimator

A basic condition for the use of OBD is that the saliencies (10)-(11) give reliable estimates of "true" saliencies, *i.e.*, the increase in cost function when deleting a weight. In Figure 8 we test the validity of the two approximations entering the estimate of the individual saliencies. The experimental (true) saliency is plotted versus the estimated saliency at four different stages of the pruning process for the input-to-hidden weights. The correlation is quite remarkable for the network shown, and the result is reproducible when using the pseudo-Newton optimization. Since the uncertainty of the estimates is much smaller than the range of saliencies, it never happens that weights with "large" saliencies are deleted prematurely.

We note that the uncertainty is largest for the very small and the very large saliencies. For the small weights the imprecision is due to numerical problems, while for the large saliencies it is due to the non-linearity of the network. The largest error of the saliency estimate for large saliencies is an overestimate by a factor of 2.7. In Figure 9 the true cost function and the estimated cost function are shown for two different weights with large saliencies and significant errors. The second order approximation used by OBD (and by OBS) does not give an accurate description of the cost function. Note that these weights are exceptions to the general behav-

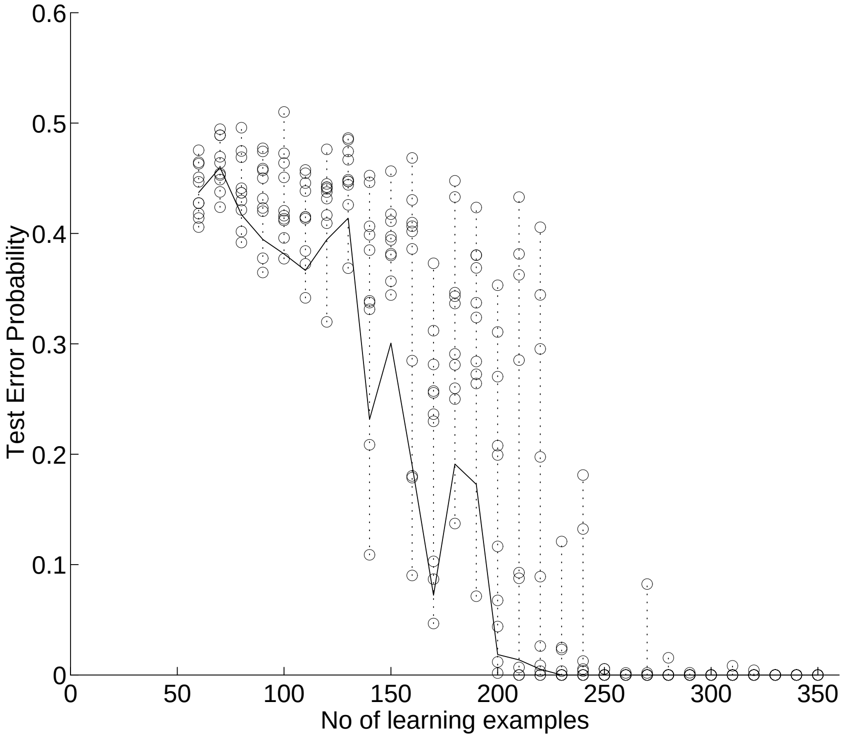

Figure 4: Learning curve of a fully connected architecture (10-12-1). At each training set size (vertical dotted lines) ten networks have been trained on the same random training set. Their test errors are indicated by circles. The *majority* vote of the ensemble is shown by the line.

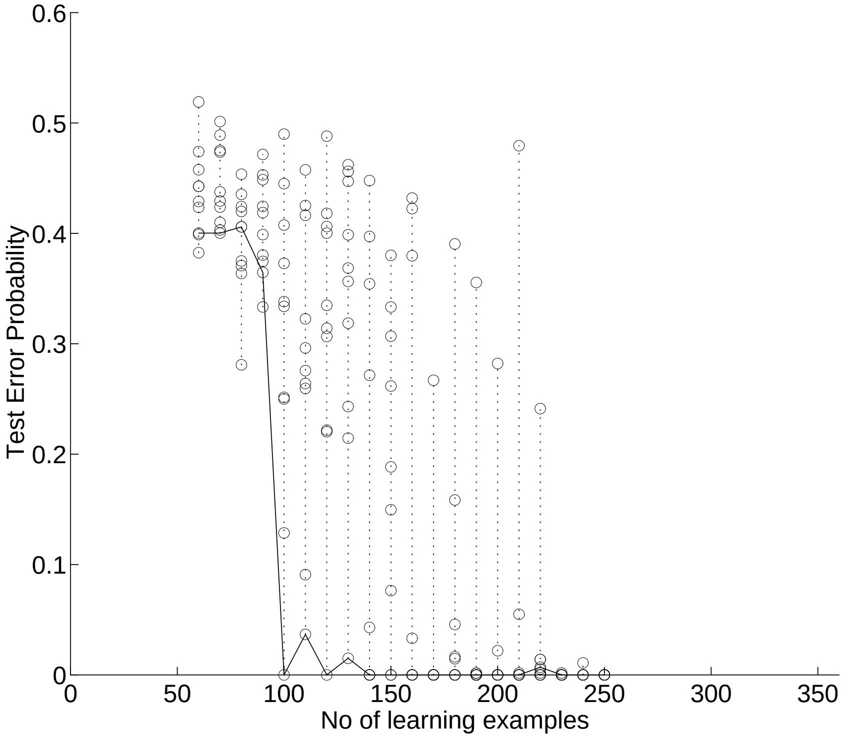

Figure 5: Learning curve for networks trained and pruned *on the run* by Optimal Brain Damage. At each training set size (vertical dotted lines) ten networks have been trained as indicated by circles. Picking the smallest network among the ten leads to test errors as shown by the line.

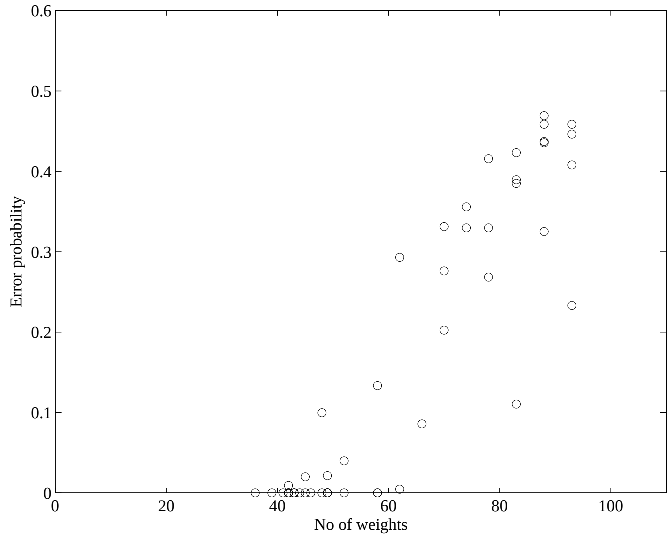

Figure 6: Probability of test error versus number of weights in a pruned network. Fifty networks were trained and pruned on a training set of size $p = 140$.

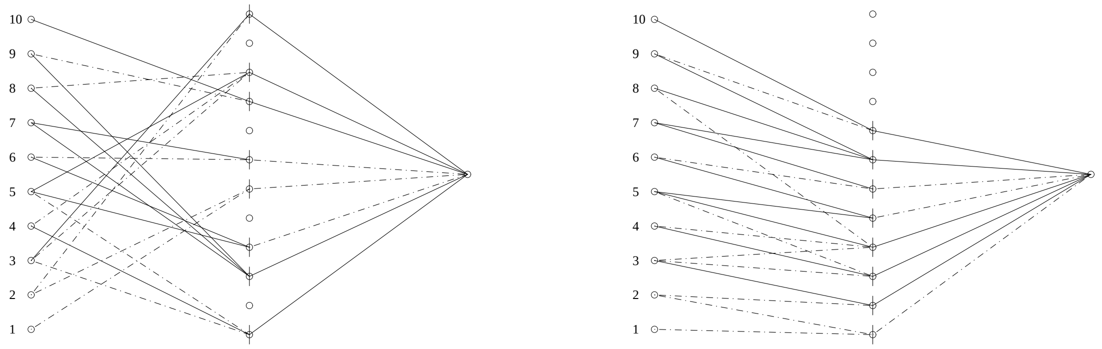

Figure 7: Optimized architecture trained on 160 randomly chosen examples. The network as it dropped out of the simulator is shown to the left. The network to the right has been unfolded with respect to receptive fields by a permutation of the hidden units. Solid lines indicate positive weights, dash-dotted lines indicate negative weights. A vertical line through a unit indicates an active threshold

ior. For the majority of weights the estimate is accurate enough to ensure correct ranking.

The experiment has been repeated with a straight back-prop optimizer. In this case the *small* saliencies are somewhat less accurately estimated, presumably because the gradient is not exactly zero at the "local minimum". We conclude, for the present network, that the effect of single weight pruning is well estimated by the saliency. Note that the off-diagonal terms do not enter the saliency for single weight deletion,

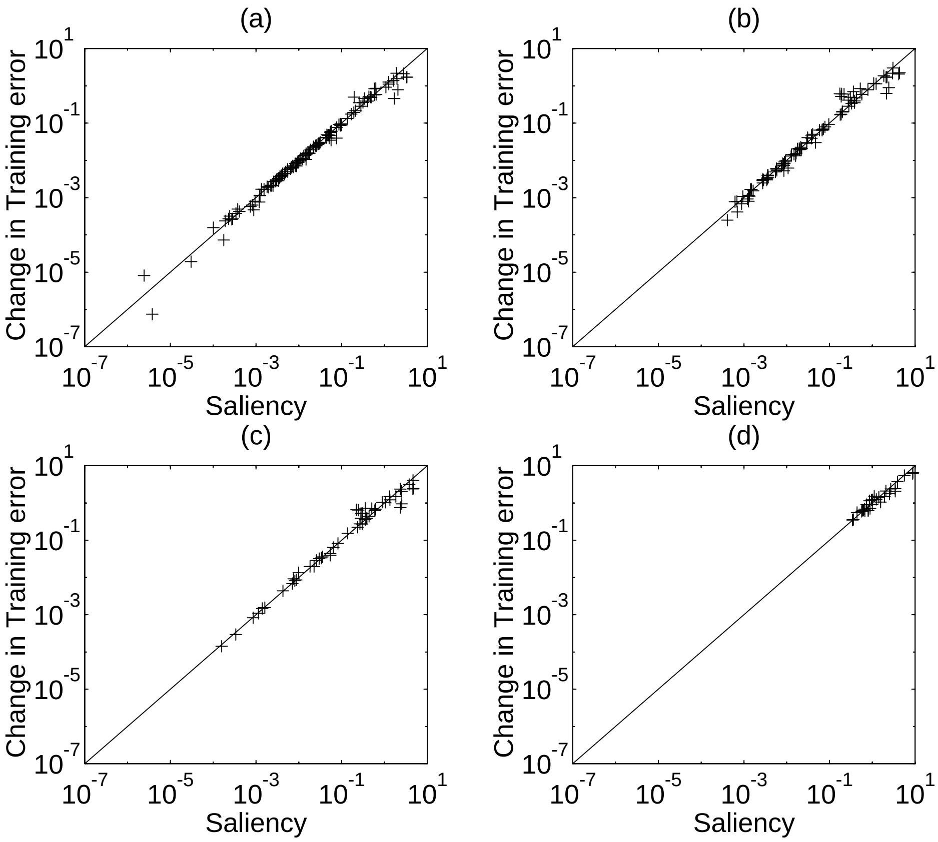

Figure 8: Experimental test of OBD approximations on the contiguity problem at four different stages in the pruning procedure: a) fully connected (145 weights), b) 104 weights, c) 66 weights, and d) 42 weights. The training set consists of 200 patterns. Only input-to-hidden weights are shown.

hence Figure 8 is testing the second order approximation to the cost function in the given minimum and the success of the optimizer in locating it.

It is surprising to see that most of the weights of this network, have saliencies that can be computed within the second order approximation. Inspecting equations (10)-(11) we note that the saliencies could be small due to the saturated activities (the hidden units saturate if the input-to-hidden weights are of large magnitude), so a small saliency would not always mean that the weight is unimportant. For large weights (and no weight decay) the error surface can be like a "bath tub" with steep walls and low curvature at the bottom. This kind of surface is poorly described by a quadratic approximation. The weight decay regularizer seems to prevent the system from entering the flat and non-quadratic bottom of the "bath tub" leading to good agreement between the measured and estimated saliencies in most cases.

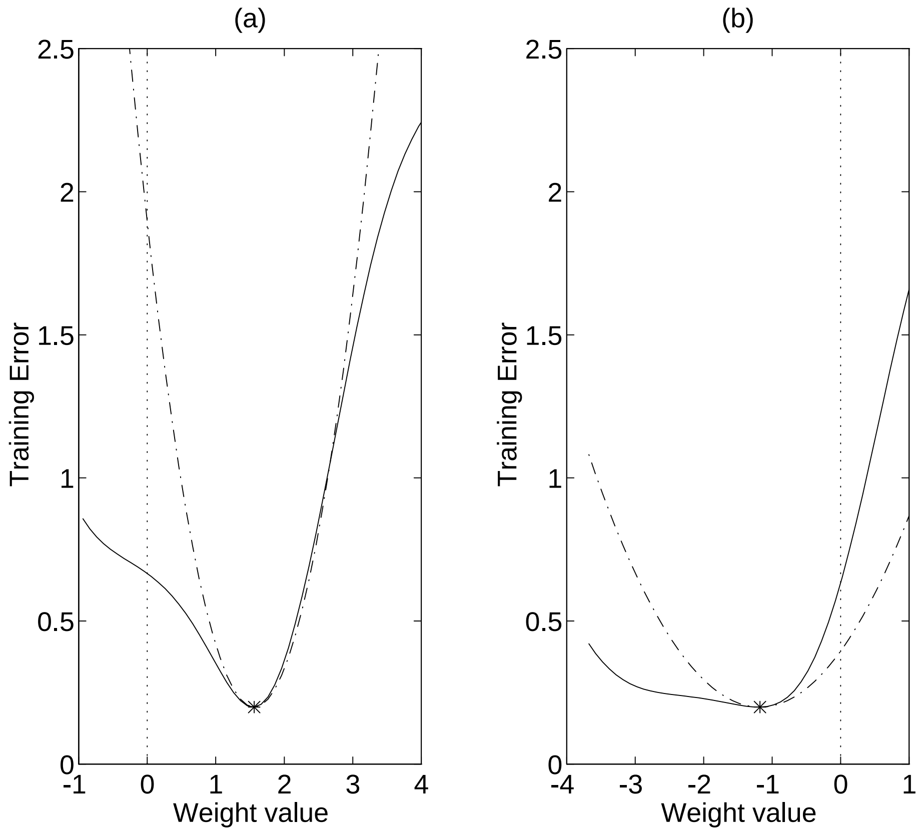

Figure 9: Test of the second order approximation for two weights with high saliency and a significant error of the saliency estimate. The second order approximation of the cost function is shown as a dash-dotted line, while the solid line represents the actual error as function of the given weight. In a) the estimate is overestimated by a factor of 2.7 and in b) the saliency is underestimated by factor of 0.6. We note that the cost function is poorly approximated by the second order approximation for both of these weights.

4.5 Experiments with the "monks problem"

The method has been tested on a few other Boolean problems. In particular the "first monk problem" [18], also used in the OBS experiments [7].

The first monk problem (*monk1*) [18] was formulated within an artificial robot domain in which robots are described by six different attributes. Back-propagation with weight decay showed the best performance among the considered classifiers [18]. The *monk1* problem can be represented by a net with 17 inputs and one output, and is defined by a total number of 432 legal examples. In [18] the training set size was 124 examples and the number of hidden neurons was 3. With this training set size Hassibi and Stork [7] report a net with 14 connections, while Lautrup [4] found a net with 14 connections using back-prop with weight decay and magnitude based pruning. For the same specifications, using back-prop, weight decay and a non-linear output we have been able to find a net with only 11 connections.[1] The network is shown in Figure 10 below; note that all thresholds are pruned away. Solutions with only two hidden units but more connections (down to 14) have been found as well.

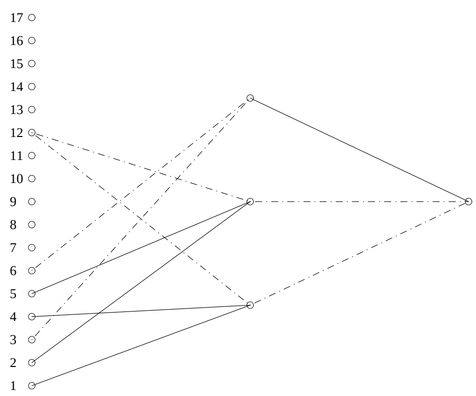

Figure 10: The network for *monk1* with 11 connections. The input-to-hidden weights have magnitudes about 0.8. The hidden-to-output weights have magnitudes about 1.6. Solid lines indicate positive weights, dash-dotted lines indicate negative weights.

5 Concluding remarks

We have investigated the Optimal Brain Damage scheme for pruning of feed forward networks. In particular we have studied the problem of learning a Boolean function from examples. Since our test problem is known to be solvable by a sparsely connected network, a fully connected network is highly redundant and a pruning procedure should be able to improve generalization. Our numerical experiments confirm this convincingly by showing that the learning curve of the iteratively trained and pruned network is significantly improved relative to the learning curves of fully connected networks. Furthermore we have shown that it is possible to improve the learning curve even more by chosing the smallest network among a small ensemble of pruned solutions. This ensemble search heuristic provides a learning curve with a remarkably sharp learning transition, much sharper than that of the fully connected architecture.

A qualitative understanding of the success of OBD follows from the observation that some of the underlying approximations are well satisfied. We believe that our careful optimization scheme and the use of weight decay play important roles in achieving these results.

[1] In response to our results, David Stork has informed us that *monk1* networks with 11 connections have been found with OBS in a new series of experiments.

Although OBS [7] uses a more sophisticated technique for ranking of weights, both OBD and OBS rely on the quadratic approximation. Future studies will have to determine whether the computational overhead is used better on OBS, on ensemble search with more OBD networks, or on strategies that go beyond the second order approximation.

It should also be mentioned that pruning is not necessarily the best way to optimize the network architecture. Much work has been devoted to incremental algorithms that build a network by gradually adding units. We believe that both constructive and destructive methods are important, and should eventually be combined in order to identify models with good generalization at a minimal computational cost.

Acknowledgements

We thank Jan Larsen, Benny Lautrup, Sara Solla, and David Stork for valuable discussions. We thank David Stork for providing us with preprints of the work on the Optimal Brain Surgeon method. Sara Solla is gratefully acknowledged for many comments on the manuscript. This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center (CONNECT).

References

[1] H. Akaike, *Fitting Autoregressive Models for Prediction.* Annals of The Institute of Statistical Mathematics **21**, 243-347 (1969).

[2] S. A. Barton, *A Matrix Method for Optimization a Neural Network* Neural Computation **3**, 450-459 (1990).

[3] E. B. Baum and D. Haussler, *What Size Net Gives Valid Generalization*, Neural Computation **1**, 151-160 (1989).

[4] B. Lautrup, *private communication.*

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

[6] L.K Hansen and P. Salamon, *Neural Network Ensembles*, IEEE Transactions on Pattern Analysis and Machine Intelligence **12**, 993-1001 (1990).

[7] B. Hassibi and D. G. Stork: *Second Order Derivatives for Network Pruning: Optimal Brain Surgeon.* Preprint, NIPS 1992, in press (1992).

[8] B. Hassibi, D. G. Stork, and G. J. Wolff, *Optimal Brain Surgeon and General Network Pruning*, in Proceedings of the 1993 IEEE International Conference on Neural Networks, San Francisco (Eds. E.H. Ruspini *et al.*) pp. 293-299 (1993).

[9] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison Wesley, New York (1991).

[10] E. Levin, N. Tishby and S.A. Solla, *A Statistical Approach to Learning and Generalization in Layered Neural Networks*, in Proc. 2nd Ann. Workshop on Computational Learning Theory (COLT'89), eds. R. Rivest, D. Haussler, and M.K. Warmuth. San Mateo, CA: Morgan Kaufmann, pp. 280-295 (1989), and Proceedings of the IEEE **78** 1574 (1990).

[11] Y. Le Cun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jakel: *Handwritten Digit Recognition with a Back-Propagation Network*, in Advances in Neural Information Processing Systems II (Denver 1989) ed. D.S.Touretzsky. San Mateo: Morgan Kaufman, pp. 396-404 (1990)

[12] Y. Le Cun, J.S. Denker, S.A. Solla: *Optimal Brain Damage*, in Advances in Neural Information Processing Systems II (Denver 1989) ed. D.S.Touretzsky. San Mateo: Morgan Kaufman, pp. 598-605 (1990)

[13] D.B. Schwartz, V.K. Samalam, S.A. Solla, and J.S. Denker, *Exhaustive Learning*, Neural Computation **2**, 371-382 (1990).

[14] S.A. Solla: *Capacity Control in Classifiers for Pattern Recognizers*, in 'Neural Networks For Signal Processing'; Proceedings of the 1992 IEEE-SP Workshop, (Eds. S.Y. Kung, F. Fallside, J. Aa. Sørensen, and C.A. Kamm), IEEE Service Center, Piscataway NJ, pp. 255-266 (1992).

[15] C. Svarer, L.K. Hansen, and J. Larsen, *On Design and Evaluation of Tapped-Delay Neural Network Architectures*, in Proceedings of the 1993 IEEE International Conference on Neural Networks, San Francisco (Eds. E.H. Ruspini *et al.*), pp. 46-51 (1993).

[16] H.H. Thodberg, *Improving Generalization of Neural Networks Through Pruning.* Int. Journal of Neural Systems **1**, 317-326 (1991).

[17] H.H. Thodberg, *The Neural Information Processing System used for Pig Carcasses Grading in Danish Slaughterhouses*, preprint 1989, The Danish Meat Research Institute, Denmark.

[18] S. B. Thrun *et al.*, *The MONK's Problems - A performance comparison of different learning algorithms.* Technical Report, CMU-CS-91-197 Carnegie-Mellon U. Department of Computer Science, 1991.

[19] L. Valiant, *A theory of the learnable*, Communication of the ACM **27**, 1134-1142 (1984).