A Comparative Study of Neural Network Optimization Techniques

Thomas Ragg, Heinrich Braun and Heiko Landsberg

University of Karlsruhe, 76128 Karslruhe, Germany,
email: {ragg,braun,landsber}@ira.uka.de

Abstract. In the last years we developed ENZO, an evolutionary neural network optimizer which surpasses other algorithms with regard to performance and scalability. In this study we compare ENZO to standard techniques for topology optimization: Optimal Brain Surgeon (OBS), Magnitude based Pruning (MbP), and to an improved algorithm deduced from OBS (unit-OBS). Furthermore we compare results to a newly proposed constructive algorithm called FlexNet.
The algorithms are evaluated on several benchmark problems, two of them from the proben1 benchmark collection. We conclude that using an evolutionary algorithm as meta heuristic like ENZO does is currently the best available optimization technique with regard to network size and performance. We show that the time complexity of ENZO is similar to Magnitude Based Pruning and unit-OBS, while achieving siginficantly smaller topologies. Standard OBS is outperformed concerning both the size reduction and The inferior results of constructive methods (Cascade Correlation and FlexNet) for both investigated benchmark seem to indicate, that such methods are not well suited for topology optimization.

1 Optimization techniques

Optimizing the topology of neural networks is an important task, when one aims to get smaller and faster networks, as well as a better generalization performance. Moreover, automatical optimization avoids the time consuming search for a suitable topology. Techniques for optimizing neural network topologies can be divided in destructive algorithms like Pruning or Optimal Brain Surgeon and constructive algorithms, e.g. Cascade Correlation. Our evolutionary algorithm ENZO [1, 2, 10] is a mixture of both methods, since it allows for reduction as well as for growing structures.

In this study we compare ENZO to greedy pruning techniques which incrementally eliminate weights until the learning error gets too bad: Magnitude based pruning (MbP) which eliminates in every step the smallest weight and two variants of the Optimal Brain Surgeon (OBS) which eliminates the weight (resp. unit) with the smallest increase of learning error for the quadratic Taylor approximation. The algorithms are evaluated on several benchmark problems, two of them from the proben1 benchmark collection.

We will show that ENZO surpasses Magnitude Pruning as well as unit-OBS by evolving topologies with significantly less size (number of input units, hidden units and number of weights) while using nearly the same amount of computing time. Moreover, the size of the evolved topologies is even smaller as constructed by the very time consuming incremental Optimal Brain Surgeon algorithm, i.e., we get a better network size reduction using several orders of magnitudes less computing power. Furthermore, the results of the destructive methods are compared to constructive methods published in [7] for the spiral problem and the cancer benchmark from the proben1 collection [8].

The main criteria for optimizing the network topology is the size of the network, furthermore the time needed for the optimization and the classification error. A problem of pure network reduction algorithms lies in the fact, that the smallest network achieving a learning error below a given error limit has not the best generalization performance in general. The tradeoff between

network size and generalization capability can be balanced by ENZO using both criteria in the fitness function while MbP and OBS do not consider the generalization and achieve therefore worse generalizing networks. In order to keep the comparison clear, we do not evaluate this essential advantage of ENZO. Therefore, the only criteria is the size of the achieved neural network under the constraint that the learning error (and thereby the classification error) remains under a given (small) error bound.

The size of the network is measured by three parameters: Number of input units, number of hidden units and number of weights. In typical applications of the multilayer perceptron model there is some redundancy in the representation (encoding of the input) in order to facilitate the learning problem, i.e., some features are precomputed and given as additional inputs, such that these features need not to be learned as hidden features. Thus we may state, that there may be a tradeoff between number of input units and number of hidden units. On the other hand, redundant or even irrelevant input units may decrease the generalization capability. Therefore, we are interested in both: Reducing the input size and the network size. The first gives hints about the salient parameters in the input representation, the second speeds up the network evaluation and both together improve the generalization capability.

1.1 Destructive techniques

All destructive techniques are used in the same way. The networks are trained until they reach a local minima, i.e., the mean square error is below a given bound or a maximal number of epochs is exceeded. A candidate c (either a link or a unit) is deleted and then the network gets retrained. If it is not possible to learn the patterns with the given constraints (#epochs,mean square error), optimizing stops and the network before the last operation is restored. Different strategies are used to select the candidates:

Pruning: Magnitude based pruning (MbP) [5] searches the weight with the smallest absolute value. This weight is the candidate to delete.

Optimal Brain Surgeon: Optimal Brain Surgeon (OBS) was introduced by Hassibi and Stork [4]. It uses the Hessian matrix to compute the less important weight, i.e., the one which causes the lowest increases in error (ΔE), if pruned. This is the weight $\mathbf{w_q}$ which minimizes $\frac{1}{2}\Delta\mathbf{w}\mathbf{H}\Delta\mathbf{w}$ such that $(\Delta\mathbf{w})_\mathbf{q} = -\mathbf{w_q}$.

A special case of OBS is Optimal Brain Damage [3], which makes the assumption that the Hessian matrix is diagonal, which causes it to delete wrong weights sometimes. Thus it is suboptimal to OBS and is not further considered here, also since results were not better than for magnitude pruning [13].

Optimal Brain Surgeon with units: Since OBS is a very expensive optimization technique, a promising variant to speed up the algorithm is to prune units or several weights at a time. Oftenly during optimization several unit loose all their connections and are removed, which makes it reasonable to start optimization by unit pruning, followed by standard OBS to eliminate weights. Unit-OBS and other variants are described in detail in [12]. Deleting several weights at once leads to a generalized equation for $\mathbf{w_{q_1}}\mathbf{w_{q_2}}\ldots\mathbf{w_{q_m}}$ minimizing a term similar as for OBS such that $(\Delta\mathbf{w})_{\mathbf{q_i}} = -\mathbf{w_{q_i}}, i = 1\ldots m$.

1.2 Constructive techniques:

Several network construction algorithms were proposed so far, from which cascade correlation might be the most popular. We compare our results on the cancer benchmark to lately published

results of the FlexNet algorithm including results for cascade correlation. Both algorithms start out by only the input and output layer. Two phases are distinguished: Firstly the network is trained until error stagnation or satisfactory performance, which stops the procedure. Secondly candidate units (or sets of candidate units) are trained, and the best one is installed in the network. FlexNet also allows for several connection strategies.

All constructive techniques have the main disadvantage of resulting in larger networks, sometimes with several hidden layers and more parameters. This leads to the consequence of over-fitting the data easily which would require more training patterns to provide a sensible generalization ability.

2 Evolutionary Algorithm

The basic principles of evolution as a search heuristic may be summarized as follows. The search points or candidate solutions are interpreted as individuals. Since there is a population of individuals, evolution is a multi-point search. The optimization criterion has to be one-dimensional and is called the fitness of the individual. Constraints can be embedded in the fitness function as additional penalty terms. New candidate solutions, called offsprings, are created using current members of the population, called parents. The two most used operators are mutation and re-combination (crossover). Mutation means that the offspring has the same properties as its single parent but small variations. Whereas re-combination (crossover) means that the offspring's properties are mixed from two parents. The selection of the parents is randomly but biased, preferring the fitter ones, i.e. fitter individuals produce more offsprings. For each new inserted offspring another population member has to be removed in order to maintain a constant population size. This selection is usually done according to the fitness of each member (Survival of the fittest).

Every heuristic for searching the global optimum of difficult optimization problems has to handle the dilemma between exploration and exploitation. Priorizing the exploitation (as hill-climbing strategies do) bears the danger of getting stuck in a poor local optimum. On the other hand, full explorative search which guarantees to find the global optimum, uses vast computing power. Evolutionary algorithms avoid getting stuck in a local optimum by parallelizing the search using a population of search points (individuals) and by stochastic search steps, i.e. stochastic selection of the parents and stochastic generation of the offsprings (mutation, crossover). On the other hand, this explorative search is biased towards exploitation by biasing the selection of the parents preferring the fitter ones.

3 Comparing the strategies for minimizing the topology

In the following we want to compare the different strategies of the four destructive algorithms (MbP, OBS, unit-OBS and ENZO) for minimizing the network topology. Comparing MbP and OBS we may state that both select single connections for elimination. Whereas MbP uses a simple heuristic (select the smallest weight), this selection is done much more carefully by OBS: It approximates the error function by a taylor approximation of second degree (quadratic polynomial) and computes for this approximation the optimal connection, i.e. the connection with smallest increase of learning error caused thru its elimination. As a positive side effect, the according weight matrix is also computed which achieves the minimal increase of learning error on the Taylor approximation. Using this weight matrix as initialization for training reduces the number of necessary training steps drastically.

A problem of this single weight elimination steps is its short sightedness: It cannot handle the synergetic effect of eliminating several connections in one step but only the singular effect of eliminating just one weight. This is improved by unit-OBS, where the effect of eliminating whole groups of connections is evaluated. Since the number of different groups of size k explodes

(cf. binomial distribution) we have to reduce the search space. In unit-OBS the search space is reduced to the groups each defined as the connections of a neuron. This more global view enables unit-OBS to achieve significantly smaller topologies in comparison to standard OBS (cf. section 6).

A problem of all three algorithms (MbP, OB, unit-OBS) is there greediness. It is not possible to backtrack an unfavorable elimination. Because there are many single elimination steps any error may cut off favorable areas of the search space. For OBS and unit-OBS there are two sources of errors:

1. The approximation error caused by the Taylor Approximation.
2. The short sightedness: Eliminating incrementally connections or units with smaller increase of learning error does not necessarily lead to a topology (of a given size) with smallest increase of learning error.

Both problems can be handled by ENZO: Because it does not only eliminated the most promising unit (or connection) but λ promising variants (mutated offsprings) and evaluates their actual increase of learning error, the approximation error of the Taylor approximation is solved. Moreover, the short sightedness is reduced by following not only one line of development but a whole population of promising lines which are extended by chance according to their fitness. This is the reason why we could expect the best results by ENZO (cf. section 6).

4 Time complexity

All four algorithms (MbP, OBS, unit-OBS and ENZO) proceed in the same way: They produce iteratively an offspring from a parent (using the slang of evolutionary algorithms) by deleting a unit or some connections (mutation) and optimize it by gradient descent (Rprop). The time complexity for producing an offspring can be subdivided in the time complexity for the mutation and for the training.

The time complexity for the mutation is determined by the selection operator which specifies the unit or connections for elimination (or addition using ENZO, respectively). For MbP and ENZO, the time complexity of selection is $O(n)$ (with n = number of connections in the topology). Since one learning step has time complexity $O(pn)$ (with p = number of learning patterns), the time complexity for training majorizes the selection and therefore the time complexity for producing one offspring is $O(epn)$ (e = number of learning steps).

For OBS and unit-OBS the time complexity of the selection operator is $O(pn^2)$ (cf. [4, 12]). Therefore it majorizes the time complexity of training ($O(epn)$) if $e < n$. This is true (on average), since both OBS and unit-OBS compute not only the connection (or unit respectively) with the smallest increase of the learning error (thru its elimination) but also the weight matrix, which achieves this smallest learning error. The only reason, why there are nevertheless some learning steps necessary, is caused by the approximation error of the quadratic polynomial (Taylor approximation of second order, - if this approximation would be exact, then learning may be skipped). In our experimental investigations the number of learning steps were typically less than ten. Therefore we may conclude, that the time complexity of producing an offspring is $O(pn^2)$ for both OBS and unit-OBS.

Comparing the time complexity for all four algorithms, we have to take into account the number of offsprings which are produced. Of course, this depends upon how many units or connections can be eliminated. On a minimal topology none of the algorithms can produce an improved offspring. In typical applications, however, the number of connections can be drastically reduced to less than the half. Therefore we may assume, that the number of eliminated connections is O(n).

The elimination of a unit can be simulated by the elimination of all its connections. If we consider a weight matrix with $n = mm$ weights, than the elimination of a unit is the

elimination of its according row and column, i.e. about $2m$ weights are eliminated. Therefore a unit elimination can be simulated by the elimination of $O(\sqrt{n})$ connections. Consequently, we may conclude, that we need $O(\sqrt{n})$ eliminations of units.

Both ENZO and unit-OBS eliminate first units and then connections. Since the time complexity is proportional to the size of the network, the time consumed for producing the first offsprings is much larger than the time for eliminating single connections when the topology is already nearly minimal (such that no unit can be eliminated). Therefore, the time complexity is dominated by the elimination of units and the elimination of connections may be neglected for both unit-OBS and ENZO.

In our parameter setting of ENZO, in every generation the number of units is decreased by about one, so that we get about the same number of generations as the number of unit-elimination steps by unit-OBS. The only difference lays in the fact, that ENZO produce in each generation λ offsprings whereas unit-OBS only one per elimination step. Summarizing we get for all four algorithms the following time complexities:

MbP	OBS	unit-OBS	ENZO
$O(epn)O(n)$	$O(pn^2)O(n)$	$O(pn^2)O(\sqrt{n})$	$O(epn)O(\lambda\sqrt{n})$
$= O(epn^2)$	$= O(pn^3)$	$= O(pn^{2.5})$	$= O(\lambda epn^{1.5})$

Table 1: The table gives the time complexity of the four examined optimization algorithms. e is the number of training epochs, p equals the number of patterns, n the number of weights and λ the number of offsprings.

From this overview we may conclude, that unit-OBS outperforms OBS by a factor $O(\sqrt{n})$. This theoretical analysis is consistent with our experimental investigations where we got a speed up of 20 for an initial topology with about 400 connections. If we compare the remaining three algorithms (MbP, unit-OBS, ENZO) pairwise the relative time complexity depends on the number of training epochs e and the number of offsprings λ. E.g. if we compare unit-OBS with ENZO and neglect the common factors of the time complexity it remains the factor $O(\lambda e)$ for ENZO and $O(n)$ for unit-OBS. In our experimental investigations we set λ to 3 and measured about the same cpu-time for unit-OBS and ENZO.

Comparing ENZO and MbP and neglecting also the common factors in the time complexity it remains the factor $O(\lambda)$ for ENZO and $O(\sqrt{n})$ for MbP. This suggests a small time advantage for ENZO, because the number of offsprings λ is only ten or less. Nevertheless MbP was in our experimental investigations a little bit faster than ENZO. The reason of this unexpected result lays in the fact, that the number of used training epochs e is significantly smaller for eliminating only a weight in comparison to eliminating an unit.

5 Generalization behavior

It is well known that the generalization capability may decrease thru intensively optimizing on the learning set (overfitting). This overfitting effect can be handled by limiting the degree of freedom of the network topology. There are two types of limitations: Firstly we can reduce the number of free parameters (weights) and secondly we may penalize the size of these parameters by a weight decay factor. Both approaches are important for a good generalization behavior. The reduction of the number (of weights) is automatically obeyed since we reduce the network topology. The penalization of the size (weight decay) has to be optimized beforehand. The weight decay factor has to be choosen so large, that the overfitting effect is just suppressed. A larger weight decay factor decreases again the generalization behavior, because in this case the gradient descent priorizes the minimization of the size of the weights instead of the learning error.

Therefore, we optimized in our experimental investigations the weight decay factor beforehand according to the above considerations. In Fig. 1 we see that in spite of this optimal tuning of the weight decay factor, the generalization behavior improves significantly during the reduction of the topology. This improvement is mainly due to the elimination of redundant input units. By that, the learning algorithm can not optimize the error function on the learning set by using irrelevant input features and has to concentrate on the relevant ones.

a) b)

Fig. 1. The generalisation error on the test set for the thyroid gland benchmark is plotted vs. optimization steps for a) unit-OBS: The three different phases (deleting of input units, hidden units and weights) are plotted with different line styles. b) ENZO: Worst, average and best error for each generation.

6 Results

We compared the optimization techniques on several benchmarks, i.e., two simple classification problems, spirals and TC, as well as two real world problems, thyroid glance from and breast cancer classification. For all benchmarks we used fully-connected networks with shortcut connections between around all layers. As learning algorithm we used Rprop [9] with weight-decay. If not stated otherwise the parameters of Rprop were set to 0.01 for the initial step size, 20.0 for the maximal step size. The weight decay is adjusted manually in a series of training runs by increasing its value until overfitting on the test set does not occur anymore. The activation function was the standard sigmoid function $(1/1+e^{-x})$ for all units. Initialization of the weights was done in the range from -0.5 to 0.5. The number of generations, the population size and the number of offspring each generation for ENZO are 30, 9 and 3, respectively.

6.1 The TC-Problem

The purpose of using an artificial benchmark like the TC-Problem, is solely to examine if the optimization algorithm is able to find a minimal topology that still solves the problem with a 100% correctness and the probability to do so, i.e., to determine the dependency from the initialization. The training set consists of all possible T's and C's on a 4x4 pixel matrix, i.e. 17 T's and 23 C's.

	Pruning	OBS	unit-OBS	ENZO
median topology	14-5-1	13-5-1	11-4-1	9-2-1
best topology	12-2-1	12-4-1	8-2-1	8-2-1
median #weights	33	29	23	22
best #weights	20	22	22	18

Table 2: The table shows the results of the optimization processes for the TC-Problem. The first two columns give the results for the median from 21 runs, i.e., the 11th best run. The other two rows are the values for the best run. The weight-decay was 10^{-5} for all algorithms.

The topology is 16-16-1, an input unit for each pixel with 288 weights in total. Table 2 shows that only unit-OBS and ENZO were able to determine the minimal topology, where the average results of ENZO are still better.

6.2 Spirals

This benchmark is also artificial, but interesting since it is highly non-linear and difficult to learn. Also the generalization ability, interpolation and extrapolation, of networks can be tested to some extent. Our data set consists of 194 training patterns and 576 test patterns. The topology is 2-5-5-5-1, with 122 weights.

	Pruning	OBS	unit-OBS	ENZO
median topology	2-5-5-5-1	2-5-5-5-1	2-5-5-4-1	2-4-4-4-1
median #weights	122	122	90	70
best topology	2-5-5-5-1	2-5-5-5-1	2-4-3-3-1	2-4-3-3-1
best weights	89	63	50	48

Table 3: The table shows the results of the optimization processes for the spirals problem using a weight decay of 0.1. See also caption of table 2.

The input are the x-y-coordinates, the output classifies the color, either black or white. We used a high weight decay of 0.1, and 10^{-4} resp. 10^{-3} as step size parameters for Rprop. This leads to convergence after 3000 to 5000 training epochs. The classification error for all algorithms varies from 1.7% to 3%. This is much less than reported for cascade correlation (9.3%) and FlexNet (6.1%). This results depend heavily on the high weight decay and the low maximal step size used for Rprop.

6.3 Breast Cancer

Breast cancer is a real world benchmark from the proben1 collection [8]. It was originally obtained from the University of Wisconsin [6]. A tumor is to be classified as benign or malignant based on cell description, e.g., the cell size and shape, gathered by microscopic examination. The network topology is 9-8-4-2 with weights.

	Pruning	OBS	unit-OBS	ENZO
median topology	8-6-2	8-3-2	5-2-2	3-2-2
median #weights	29	29	17	10
best topology	7-6-2	8-3-2	5-1-2	3-1-2
best weights	26	26	12	8

Table 4: The table shows the results of the optimization processes for breast cancer classification. See also caption of table 2

The data consists of 350 training and 349 test patterns, from which 65.5% are benign. Unlike reported in [7], we needed only about 700 training epochs to get satisfactory results and a classification error between 2% and 3%. We used a weight decay of 10^{-2}. Other parameters were set as above.

6.4 Thyroid Gland

Thyroid gland diagnostic is a also real-world benchmark from proben1. This task requires a very good classification, because 92% of the patterns belong to one class, thus useful networks should have a classification error of less than 2%. The data set consists of 3772 training patterns and 3428 test patterns. The topology is 21-10-3, with 304 weights.

	Pruning	OBS	unit-OBS	ENZO
median topology	7-10-3	9-6-3	8-6-3	6-1-3
median #weights	45	42	29	21
best topology	6-10-3	7-4-3	6-3-3	5-1-3
best weights	26	26	16	15

Table 5: The table shows the results of the optimization processes for thyroid gland classification. See also caption of table 2.

The classification error was for all methods between 1% and 2%, which is in the same range as reported in [11] for an evolved network (284 weights, 1.4% classification error) . The smallest network was evolved by ENZO using 15 weights achieving a classification error of 1.0%.

7 Discussion

Optimization techniques were compared for several benchmarks with regard to their performance in minimizing the network size. Our results show that standard techniques depend heavily on the initialization of the network, as well as on the weight decay term. On the one hand,

using weight decay makes it easier to prune minor important weights and thus leads to better results. Also the dependence on the weight initialization is decreased. On the other hand weight decay decreases the degree of freedom for fitting the weights and thereby may increase the minimal network size.

The very time consuming Optimal Brain Surgeon algorithm is outperformed by unit-OBS with respect to network size reduction and computing time. Moreover, our results show that ENZO surpasses both Magnitude Pruning and unit-OBS by evolving topologies with significantly less size while all algorithms have about the same time complexity.

Results of constructive algorithms reported in [7] were worse on the spirals and cancer benchmark, with regard to network size and performance as well as training epochs. The classification error was about three times higher.

We conclude that using evolutionary algorithms as a meta heuristic as done in ENZO is concurrently the best available optimization method, with regard to network size and performance. Furthermore, it is scalable in time and simple to parallelize. Additional constraints on network topology or performance are easily integrated into the fitness function. Thus evolution combined with learning appears as the best framework to train neural networks.

References

1. Heinrich Braun. On solving travelling salesman problems by genetic algorithms. In *Parallel Problem Solving from Nature,PPSN I*, Lecture Notes in Computer Science 496, pages 129–133, 1991.
2. Heinrich Braun and Joachim Weisbrod. Evolving feedforward neural networks. *Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms*, 1993.
3. Y. L. Cun, J.S. Denker, and S.A. Solla. Optimal Brain Damage. In *NIPS 2*, 1990.
4. Babak Hassibi and David G. Stork. Second order derivatives for network pruning: Optimal Brain Surgeon. In *NIPS 4*, 1992.
5. John Hertz, Anders Krough, and Richard G. Palmer. *Introduction to the theory of neural computation*, volume 1 of *Santa Fe Institute, Studies in the sciences of complexity, lecture notes.* Addison-Wesley, 1991.
6. O. Mangasarian and W.Wolberg. Cancer diagnosis via linear programming. *SIAM News*, (5), 1990.
7. K. Mohraz and P. Protzel. FlexNet - A Flexible Neural Network Construction Tool. In *to appear: European Symposium on Artificial Neural Networks (ESANN), Bruges, Belgium*, 1996.
8. Lutz Prechelt. Proben 1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules. Technical Report 21/94, Universität Karlsruhe, Fakultät für Informatik, 1994.
9. M. Riedmiller and H. Braun. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In *Proceedings of the ICNN*, 1993.
10. J. Schäfer and H. Braun. Optimizing classifiers for handwritten digits by genetic algorithms. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms ICANNGA95*, pages 10–13, Orlando, 1995.
11. W. Schiffmann, M. Joost, and R. Werner. Application of genetic algorithms to the construction of topologies for multilayer perceptrons. In *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms ICANNGA93*, pages 675–682, 1993.
12. Achim Stahlberger. OBS - Ein Verfahren zum Ausdünnen neuronaler Netze. Verbesserungen und neue Ansätze. Diplomarbeit, Universität Karlsruhe, Institut fuer Logik, Komplexität und Deduktionssysteme, 1996.
13. Andreas Zell. *Simulation Neuronaler Netze*, volume 1. Addison-Wesley, 1994.

This article was processed using the LaTeX macro package with LLNCS style