# ADAPTIVE INSERTION OF UNITS IN FEED-FORWARD NEURAL NETWORKS

**David Harris** and **Tamás D. Gedeon**[1]
Department of Computer Science,
Brunel University, Uxbridge UB8 3PH, U.K.

**Abstract**:

Feed-forward networks trained by back-propagation are very popular, for reasons of simplicity, availability, and past successes. To train such networks within reasonable time scales, or to train them at all, it is necessary to include more hidden units than would appear to be required. The rule of thumb is that if the network doesn't learn, add more units. If the new units end up duplicating the functionality of existing units, we have gained nothing. Training large numbers of units is expensive, particularly the training of units which do not make a significant contribution to achieving the solution. We have made a study of the activity of hidden units, and can add units which have guaranteed distinct functionality from the other units present.

**Key words**:

neural networks; feed-forward; back-propagation; adding units; training speed

[1] To whom correspondence should be addressed.

Now at: School of Computer Science and Engineering,
The University of New South Wales,
P.O. Box 1, Kensington 2033, AUSTRALIA
Phone: +61 2 697 5526 Fax: +61 2 313 7987

**Objective**:

The aim of this work is to be able to produce small artificial neural networks with adequate functionality for any specific problem domain. To this end we will first review approaches in finding minimal networks, as well as approaches in increasing network size.

**Introduction**:

In this paper we will generally assume a feed-forward network of three layers of processing units. All connections are from units in one level to the subsequent one, with no lateral, backward or multilayer connections. Each unit has a simple weighted connection from each unit in the layer above. For simplicity of discussion only, we have assumed a single output unit separating the input patterns into two classes. The network is trained using a training set of patterns with desired outputs, using back-propagation of error measures (Rumelhart and McClelland, 1986). Most workers now use batch updating, rather than sequential pattern by pattern updating of weights. By back-propagation we mean the general concept of developing the error gradient with respect to the weights, and not restricted to the standard gradient descent method. In the examples we cite here we have used the the basic logistic activation function $y=(1-e^{-x})^{-1}$, again this is not germane to the generality of our method. Training by back-propagation is popular because of its simplicity theoretically, and the ease of use and production of such networks. There is a wide availability of simulators, often at low cost. The back-propagation method has been used successfully in a number of disparate areas ranging from speech synthesis (Sejnowski and Rosenberg, 1987), to bond underwriting (Dutta and Shekhar, 1988).

**Disadvantages of back-propagation**:

The major disadvantage of back-propagation is that it is slow. The most outstanding and a priori least solvable reason is the inability to specify the size of network required for any particular problem. This concerns especially the number of hidden unit, since the numbers of input and output units can be determined fairly straightforwardly. If there are too few hidden units, the network will never learn. Conversely, if there are too many units, it may take an inordinate amount of time to train. These two conditions have very similar symptoms, lots of presentations of patterns, but the overall error measure decreases very slowly.

We will not discuss algorithmic solutions to disadvantages of back propagation, except to note that while using weight decay, if the weight linking two units has decayed to zero at the end of training, the activation of the unit using this link is ignored and is analogous to the direct architectural removal of the unit.

The problem of deciding network size is likely to remain difficult in that if the number of hidden units required for a minimal solution could readily be calculated, then this would imply that we could decide what internal representations will be required. We would then be able to solve the problem by some method directly using these internal representations, and would not need the training-by-example capabilities of neural networks. That is, the use of neural networks in general is likely to remain most important in areas in which the then current state of knowledge there remain problems which cannot be solved directly.

Many researchers have remarked that to train networks successfully, or in a reasonable, more hidden units are required than the minimal number. Such minimal numbers can be created by brute force approaches such as eliminating random units and attempting to retrain the network, and backtracking if unsuccessful and trying another unit to remove. Minimal networks can be produced this way which cannot be trained from a random state. For simpler problems it is also possible to hand-craft minimal network solutions, which again are untrainable ab initio.

**Enlarging networks**:

In this section we will first discuss the RCE-network due to Reilly, Cooper and Elbaum

(1982) which we will consider a simplified form of the common feed-forward network trained by back-propagation method, though no such causal connection exists; the Cascade correlation method due to Fahlman and Lebiere (1990); followed by methods developed for more conventional feed-forward networks.

The architecture of the RCE-network is often described in terms of the square well limit of the N-dimensional Coulomb potential. For our purposes here, it is more useful to describe the RCE-network in terms of its architecture, consisting of three layers: the input layer, an internal layer, and a simplified processing output layer.

Each unit in the internal layer sends unit strength signals to a single unit in the output layer if the position of the test pattern in the N-dimensional feature space is within the hypersphere with a radius corresponding to the threshold of the unit, centered around the vector of its connections to the input layer.

The RCE-network learns in two distinct ways, by modifying the threshold of internal units so as to reduce the size of the N-dimensional feature recognition hypersphere, and by the addition or commitment of new units to the internal layer. Thus, RCE-networks learn by the accretion of units and their fine tuning in terms of their feature space hypersphere radius, rather than by the modification of weights.

Output units correspond to pattern categories. The network is said to have assigned a category to an input pattern if the output unit for a category fires. Incorrect firing of output units as contrasted with the expected outputs causes error signals back to the internal layer. This error signal can have two values only, corresponding to cases where the output unit was off when it should have been on (lack of recognition), and to cases where the output unit was on when it should have been on (false recognition). These signals are actually implemented as messages of ±1.

The major disadvantage of the RCE-network is that the network is particularly susceptible to the effects of the order of presentation of input in determining efficient coverage of the pattern space and generalisation. In some sense, there is no built in generalisation at all, there may be some parts of the pattern space which are not covered by the RCE-net – for these regions the net can make no decisions even if it is a region 'surrounded' by regions of a particular output class. The RCE-net could be readily implemented on a serial computer by using a sparse array, storing the real valued feature space location and radius of the the individual internal units, and the address of the output unit they serve. Clearly the advantage of the RCE-net over this simulation lies in the possibility of a truly parallel implementation. The lack of inherent generalisation, and overly deterministic behaviour, can be compensated for by the use of multiple RCE-nets in a larger system, each RCE-net making decisions on the basis of some subset of the data. An example of this is the implementation of a Mortgage Underwriting system using a multiple RCE-net system (Collins, Ghosh, and Scofield, 1988).

From the viewpoint of examining the RCE net as a solution to some of the problems of back-propagation, we are comparing apple and oranges - the RCE net learns faster, but to recover the lost generalisation capabilities we need multiple nets. Therefore, no clear statement can be made upon which is better.

The cascade correlation (Fahlman and Lebiere, 1990) architecture starts from a simple structure which consists only of the inputs linked directly to the output units, The output units are trained as far as possible, which is detected by lack of significant error reduction over some number of epochs.

Another pass is made over the training set to measure the error, and terminate the training if the residual error is sufficiently low. Alternatively, a new unit is added to the network, selected from a pool of candidate units. The pool of candidate units are trained for a number of epochs with inputs of the networks external inputs as well as the results from any existing non-output units. The new unit selected is the one whose output is best correlated with the residual error. New units are connected as they are trained to all external inputs and to the outputs of all other non-output units.

The advantages of the cascade correlation algorithm are that the majority of the network is frozen, and there is on one layer of weights being trained at any one time. This has the advantage of reducing the herd effect in that only candidate units are free to respond to error

information. Candidate units in the pool can also attempt to learn the error signal without the competition/interaction with other units, as there are no lateral connections within the pool, nor secondary 'connections' via effects on the error term as the pool is observational only, units within it cannot reduce the error measure.

Further quoted advantages are the ability to train candidates in parallel on appropriate hardware, and the strictly one way connections required in this architecture more closely approximating biological synapses, since in a hardware implementation, error signals would have to be explicitly propagated backwards. These quoted advantages gloss over significant disadvantages in their presentation phases. In a parallel hardware implementation, the sequential nature of the communication path from the first added unit to the latest added unit would be significant. This problem also makes the network topology most implausible if we are to claim any sort of rash biological comparison. Since the communication problem does not exist at a simulation level however, this method remains of interest and may be modifiable away from its inherently sequential nature. The speeds reported during simulation trials are certainly very impressive.

There are other approaches using novel topologies which we will not discuss in detail, such as the SONN model (Tenerio and Lee, 1989) which places units in a binary network and uses a simulated annealing search in deciding the usefulness and whether to retain them; and GrowNET by Smith (1991), which demonstrates that many of the conventions that have developed in the use of back-propagation can be broken and still remain with the spirit.

Dynamic node creation by Ash (1989) uses standard back-propagation on small networks and adds new units to the hidden layer when two conditions are met. Firstly, the ratio of the drop in the squared error over the last n trials compared to the squared error when the last node was added must fall below some user defined trigger value. Secondly, at least n trials must have been performed before a new unit can be added, This is required so that the computed slope of the square error is meaningful in that the same topology was involved, in the sense of the number of hidden units. New units are fully connected to previous and subsequent layers, with small random weights. It is also necessary to specify conditions relating to average and maximum squared error so as to stop adding into when the desired level of performance has been reached. When a new node is added the weights are particularly small (±0.1666), and initially takes little part in the processing of the network. The results were generally within 3 units of known minimal solutions. Statically sized nets at the minimal size at startup often take inordinate amounts of time to train as is well known. Ash reports that starting with a lower dimensionality network and then increasing its size gradually, a network can spend less time training at maximum size than the equivalently sized back-propagation model. He suggests that initial training in lower dimensional space is useful. Certainly in the reverse situation where the dimensionality is constricted, there is some smoothing and generalisation of the error surface into the lower dimension (Kruschke, 1989).

The distinctiveness of hidden units is determined from the unit output activation vector over the pattern presentation set. For each hidden unit we construct a vector of the same dimensionality as the number of patterns in the training set, each component of the vector corresponding to the output activation of the unit. This vector represents the functionality of the hidden unit in (input) pattern space (Gedeon and Harris, 1991).

The recognition of similarity of pairs of vectors is done by the calculation of the angle between them in pattern space. Since all activations are constrained to the range 0 to 1, the vector angle calculations are normalised to 0.5, 0.5 to use the angular range of 0-180° rather than 0-90°. Units with exactly identical functionality are uncommon, however, angular separations of up to about 15° are considered effectively identical. Similarly, units which have an angular separation over about 165° can be seen to be effectively complementary.

Distinctiveness analysis has been used to regularly monitor the input pattern space functional vectors of the units, to find any which are too similar. The strategy employed in the distinctiveness analysis approach is to regularly monitor the network during training. Learning is improved by the use of noise to modify any units which are not sufficiently distinct (Harris and Gedeon, 1991), this method can be used to guarantee the distinctiveness of functionality of any units added to the network.

**Stopping**:

All of the methods discussed so far have some means of restricting their application to meaningful uses, ranging from re-testing to being implicit in the method. Clearly, removing too many units can destroy the usefulness of a network, while adding too many units could degrade the speed and potentially the generalisation capacities of a network.

Ash's dynamic node creation algorithm turns off the addition of units when the training set has been learned to a user specified precision in terms of the error measure. Hirose et al use a heuristic that in the last 100 epochs if the total error decreases by less than one present another unit is added, unless the network converges. Hagiwara's badness factor algorithm is more complex, being modified for different examples, ranging from adding units after every 30 resets done on a less than 5% error reduction to adding units after every 10 resets done on less than 1% error reduction. The process terminates when the network converges. Convergence is always subject to some user specified precision. This method is risky in that an unfortunate choice in precision value may not allow the network to cease being enlarged even if addition of extra units does not reduce the error sufficiently. The net may have already learnt as well as possible the training set, thus a new unit could not reduce the error, but may increase. This can occur if by chance it is complementary to an existing useful unit, by cancelling some or all of its effect. Distinctiveness analysis does not suffer from this potential problem because the addition of units ceases, if after a unit is coerced into becoming (more) distinct, one or more units are found to be unnecessary. This is a clear sign that there are no more significant or consistent features for the network to learn, given that the error back-propagation training has caused a convergence of functionality.

Of the three methods discussed for increasing the size of networks to aid learning, and then reducing their size subsequently, there are two main approaches.

Hirose et al eliminate the last added unit assuming it will have had the least time to acquire significant functionality even though the unit was necessary to allow the network to converge. Hagiwara eliminates the unit with the highest badness factor being the largest accumulated error changes. This is the node which has done the most learning in the near past and is hence likely to be the most recently added unit. The virtue of eliminating the last unit added is not clear although it seems adequate in practice. If we look at more than the last two or three units, these methods would diverge more, however we are justified in considering them as equivalent given evidence from collective experience as well as experimentation (Ash) indicating that no more than 2 or 3 extra units are usually required to be added to a minimal network to enable ready convergence.

Gedeon and Harris' distinctiveness analysis uses a strictly functionality based approach, the last unit added is no more likely to be removed than any other for any reason other than its functionality.

**Distinctiveness and adding units**:

The following table shows a set of six hidden units with two pairs of similar units, with the units in one pair being anti-similar or complementary to either of the units in the other pair.

| | Hidden Units | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Pattern | 1 | 2 | 3 | 4 | 5 | 6 | Result | Target |
| p.000 | 0.330 | 1.000 | 0.910 | 0.582 | 0.593 | 0.381 | 0.000 | 0.000 |
| p.001 | 0.091 | 0.994 | 0.339 | 0.101 | 0.834 | 0.839 | 0.000 | 0.000 |
| p.002 | 0.098 | 0.994 | 0.348 | 0.130 | 0.874 | 0.868 | 0.000 | 0.000 |
| p.003 | 0.022 | 0.488 | 0.026 | 0.012 | 0.960 | 0.982 | 0.025 | 0.000 |
| p.004 | 0.094 | 0.994 | 0.325 | 0.128 | 0.853 | 0.849 | 0.000 | 0.000 |
| p.005 | 0.021 | 0.489 | 0.024 | 0.012 | 0.952 | 0.979 | 0.025 | 0.000 |
| p.006 | 0.023 | 0.488 | 0.025 | 0.016 | 0.965 | 0.984 | 0.025 | 0.000 |
| p.007 | 0.005 | 0.006 | 0.001 | 0.001 | 0.990 | 0.998 | 0.805 | 1.000 |
| p.008 | 0.117 | 0.994 | 0.339 | 0.132 | 0.875 | 0.870 | 0.000 | 0.000 |
| p.009 | 0.026 | 0.488 | 0.025 | 0.012 | 0.960 | 0.983 | 0.025 | 0.000 |
| p.010 | 0.029 | 0.488 | 0.026 | 0.016 | 0.971 | 0.986 | 0.025 | 0.000 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| p.011 | 0.006 | 0.006 | 0.001 | 0.001 | 0.991 | 0.998 | 0.805 | 1.000 |
| p.012 | 0.027 | 0.489 | 0.024 | 0.016 | 0.965 | 0.984 | 0.025 | 0.000 |
| p.013 | 0.006 | 0.006 | 0.001 | 0.001 | 0.990 | 0.998 | 0.805 | 1.000 |
| p.014 | 0.006 | 0.006 | 0.001 | 0.002 | 0.992 | 0.998 | 0.805 | 1.000 |
| p.015 | 0.001 | 0.000 | 0.000 | 0.000 | 0.998 | 1.000 | 0.816 | 0.000 |

The above network still has not converged as can be seen from the significant discrepancy between the result and target values for pattern number 15. It is possible to discover the pairs of similar units by inspection of the above table, but this would be prohibitively difficult to do by inspection for much larger numbers of patterns or units. The vector angles for the above table are shown below.

| Pair of units | Vector angle |
|---|---|
| 1 2 | 81.8 |
| 1 3 | 25.2 |
| 1 4 | 8.4 |
| 1 5 | 176.5 |
| 1 6 | 169.9 |
| 2 3 | 63.0 |
| 2 4 | 77.8 |
| 2 5 | 100.8 |
| 2 6 | 103.7 |
| 3 4 | 18.0 |
| 3 5 | 157.7 |
| 3 6 | 163.7 |
| 4 5 | 173.7 |
| 4 6 | 177.5 |
| 5 6 | 7.3 |

From distinctness analysis, one could decide that this network of six hidden units had sufficient units to solve the problem, since there is excess functionality as shown by the low angle between the vectors of units 1, and 4. No extra units need be added at this stage.

The following table shows part of a different network with 512 patterns, again of 6 units.

| | Hidden Units | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Pattern | 1 | 2 | 3 | 4 | 5 | 6 | Result | Target |
| . | | | | | | | | |
| . | | | | | | | | |
| . | | | | | | | | |
| p.044 | 0.812 | 0.000 | 0.741 | 0.937 | 0.011 | 0.000 | 0.997 | 1.000 |
| p.045 | 0.998 | 0.000 | 0.588 | 0.009 | 0.000 | 0.000 | 0.363 | 0.000 |
| p.046 | 0.022 | 0.000 | 0.123 | 0.008 | 0.000 | 0.000 | 0.377 | 0.000 |
| . | | | | | | | | |
| . | | | | | | | | |
| . | | | | | | | | |
| p.336 | 0.000 | 0.999 | 0.000 | 0.000 | 0.004 | 0.978 | 0.976 | 1.000 |
| p.337 | 0.041 | 0.936 | 0.000 | 0.000 | 0.000 | 0.049 | 0.006 | 0.000 |
| p.338 | 0.000 | 0.832 | 0.000 | 0.000 | 0.000 | 0.998 | 0.993 | 1.000 |
| . | | | | | | | | |
| . | | | | | | | | |
| . | | | | | | | | |

This network is even further from convergence, judging by the discrepancy between the result and target columns. All of the units look different on casual inspection, the vector angle separations are shown below.

| Pair of units | Vector angle |
|---|---|

| | |
|---|---|
| 1 2 | 78.3 |
| 1 3 | 69.1 |
| 1 4 | 82.6 |
| 1 5 | 80.5 |
| 1 6 | 89.8 |
| 2 3 | 89.7 |
| 2 4 | 79.5 |
| 2 5 | 81.5 |
| 2 6 | 75.8 |
| 3 4 | 84.5 |
| 3 5 | 87.3 |
| 3 6 | 89.9 |
| 4 5 | 79.9 |
| 4 6 | 89.8 |
| 5 6 | 88.4 |

Clearly, all the units are distinct with no redundant functionality. This is a prime candidate for the addition of at least one extra unit. The above network never does converge if left at the current size.

**Conclusion**:

We have shown that using distinctiveness analysis, we can discover in a largely algorithm independent manner when a network has insufficient functionality to solve a problem, and can add units with a guaranteed distinct functionality from the other units present. We continue this process until we discover that there is a superfluity of units, and functionality. We believe that our method adds only the number of extra units required for efficient learning, though we have not yet studied this in detail. After the network has been trained, we can use the technique of distinctiveness analysis to discover the unnecessary units, and remove them to produce a minimal network.

**References**:


Ash, T, "Dynamic node creation in backpropagation networks," ICS Report 8901, University of California, San Diego, 1989.

Collins, E, Ghosh, S, Scofield, C, "An application of a multiple neural network learning system to emulation of mortgage underwriting judgements," ICNN, 1988.

Dutta, S, Shekhar,S, "Bond rating: A non-conservative application of neural networks," IEEE Int Conf on Neural Networks, vol. II, pp. 443-450, 1988.

Fahlman, SE, Lebiere, C, "The cascade-correlation learning architecture," CMU-CS-90-100, Carnegie Mellon University, 1990.

Gedeon, TD, Harris, D, "Network Reduction Techniques," Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, 1991.

Hagiwara, M, "Novel back propagation algorithm for reduction of hidden units and acceleration of convergence using artificial selection," IJCNN, vol. I, pp. 625-630, 1990.

Harris, D, Gedeon, TD, "A meta-learning strategy to improve the performance of training algorithms," EXPERSYS-91, Paris, 1991.

Hirose, Y, Yamashita, K, Hijiya, S, "Back-propagation algorithm which varies the number of hidden units," Neural Networks, vol. 4, pp. 61-66, 1991.

Kruschke,JK, "Improving generalization in back-propagation networks with distributed bottlenecks," IJCNN, vol. I, pp. 443-447, 1989.

Reilly, DL, Cooper, LN, Elbaum, C, “A neural model for category learning,” Biol. Cybernetics, vol. 45, pp. 35-42, 1982.

Rumelhart, DE, Hinton, GE, Williams, RJ, “Learning internal representations by error propagation,” in Rumelhart, DE, McClelland, “Parallel distributed processing,” Vol. 1, MIT Press, 1986.

Sejnowski, TJ, Rosenberg, CR, “Parallel networks that learn to pronounce English text,” Complex Systems, vol. 1, pp. 145-168, 1987.

Smith, G, “GrowNet,” Personal Communication, 1991.

Tenerio, MF, Lee, W-T, “Self organizing neural network for optimum supervised learning,” TR-EE 89-30, Purdue University, 1989.