CREATING ROBUST NETWORKS

T.D. Gedeon
School of Computer Science and Engineering,
The University of New South Wales,
P.O. Box 1, Kensington, 2033,
Australia

and

D. Harris
Department of Computer Science,
Brunel University,
Uxbridge, UB8 3PH,
U.K.

Abstract:

Feed-forward networks of a few layers trained by back-propagation can be used to solve a variety of problems, however 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. Ideally, we should train the network with 'just enough' extra hidden units to minimise the training time and then remove the extra units. A disadvantage of minimal networks is that we lose the claimed robustness of neural networks in terms of survival of destruction of some parts of the network. Neural networks are always susceptible to the elimination of particular units, as this robustness is not necessarily well distributed, and is generally uncharacterisable or unquantifiable.

We have made a study of the activity of hidden units in feed-forward back-propagation networks, and can produce a minimal network which we can then retrofit with the desired robustness criteria. This approach of explicitly modifying the network to add robustness has the added advantage of allowing control over the the localisation within the network of the 'safety net' hidden units. They can then be placed well separated from the units they are protecting. This is particularly important in applications using specialised neural hardware exposed to physical damage.

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 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 or production of such networks, and the wide availability of low cost simulators.

CH 3065-0/91/0000-2553 $1.00 © IEEE

Minimal Nets:

Neural networks generally store information or knowledge in a well distributed fashion. Reducing a network to the minimum number of units may force the network to localise that knowledge, allowing for it to be extracted and used in symbolic computation. This idea of extraction of knowledge learnt by neural networks is largely unexplored, but is an interesting possible future area of interest.

The localisation of knowledge or even just the minimalisation of the number of units in the network has important and perhaps unfortunate side effects. One of the much touted advantages of neural network solutions in domains where other solutions exist is their robustness and damage resistance. Clearly, in a localised network the elimination of a single unit will cause the loss of some significant piece of information causing some significant degradation of performance. In a non-localised, but minimal network, the elimination of a simple unit will still cause noticeable degradation by definition, since otherwise we would have been wrong in labelling it a minimal network in the first place. Further, it has been suggested that minimal networks suffer some disadvantages in terms of generalisation beyond their training sets (Sietsma and Dow, 1991). This result contradicts conventional wisdom which indicates that smaller numbers of units aid generalisation. This has also been investigated by Kruschke (1989), in the context of reducing the effective number of units in terms of functionality. The difference is perhaps due to how close to the bone we cut during pruning. Sietsma and Dow's method cuts perhaps too close in reducing the number of units to levels based on the number of classes and by-inspection analyses. Our experience in the area of network reduction (Gedeon and Harris, 1991) is more in accord with the conventional wisdom, that pruned nets generalise better.

In any case, a minimal network has reduced damage resistance, and reduced robustness. This situation does have a property which non-minimal neural networks do not share, in that any damage resistance and robustness is known to be well distributed, ie close to zero. A technique to post facto add damage resistance and robustness would be very useful and could be best utilised from such a starting point.

Distinctiveness:

The *distinctiveness* (Gedeon and Harris, 1991) of hidden units is determined from the unit output activation vector over the pattern presentation set. That is, 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.

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 sufficiently similar and one of them is removed. The weight vector of the the unit which is removed is added to the weight vector of the unit which remains. With low angular separations as above, the averaging effect is insignificant, the mapping from weights to pattern space remains adequate in that the error measure is no worse subsequently.

This produces a network with one fewer unit which requires no further training. A number of individually redundant units can be removed simultaneously in this fashion. Similarly, units which have an angular separation over about 165° can be seen to be effectively complementary, and both can be removed.

The following table shows a set of six hidden units with two pairs of similar units, with the units in one pair being 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

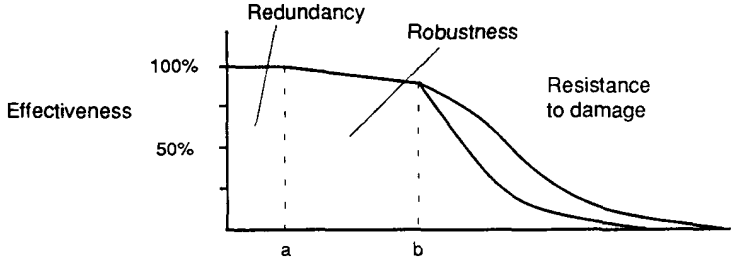
The above network still has not converged as can be seen from the significant discrepancy between the result and target values for pattern number 15. It is possible to discover the pairs of similar units by inspection of the above table, but would be prohibitively difficult for much larger numbers of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

Units *1*, and *4* are similar, as are units *5*, and 6 as shown by the low angles between their vectors. The units in each pair can be considered to provide some robustness, in the event of damage to one unit, the other would still maintain their common functionality. While there is clearly almost enough overlap in functionality between units *3* and *4* to call them similar, unit *2* has no other units with any markedly overlapping functionality. This network demonstrates the typical inhomogeneity of robustness and sensitivity to damage of feed-forward networks. Certainly, if a hidden unit was 'damaged', there is only a one in six chance of removing marked functionality by hitting unit *2*, a one in three chance of removing significant functionality by hitting one of units *3* or *4*, and a one in two chance of causing no damage by hitting any other units. Clearly, a situation where the removal of <u>any</u> unit would cause the same (minor if any) damage would be preferred.

Distinctiveness analysis has been used to analyse the functionality of units and pattern space to determine whether there are any redundant units (Gedeon and Harris, 1991), used in a meta-learning strategy to speed up learning, and to determine when new units should be added to a network (Harris and Gedeon, 1991a and 1991b).

Here we add robustness and damage resistance. We define as *robust* a system that shows minimal or no effect from some damage, and define *damage resistance* to be related to the slowness of degradation of functionality after sufficient damage has taken place that degradation is readily observable, as shown in the following diagram:



Amount of damage suffered

The area to the left of *a* is the protection offered by strictly duplicative redundancy, due to extra copies of units in the network. In the area up to *b*, the network demonstrates *robustness*, suffering zero or small diminishment of effectiveness due to damage. The area between *a*, and *b* shows the effect of primary backup units. The level of effectiveness slowly decreases because of the unavoidable loss of information due to averaging effects when the functionality of two or more units is 'redistributed' to the same number or less of backup units. The area after *b* shows significant degradation of effectiveness upon further damage. The shallowness of the slope is an indication of the *damage resistance* of the network. The upper curve shows the effect of the introduction of some secondary backup units, which subsume some of the functionalities of larger numbers of units in the network, or the functionality of some number of primary backup units.

The appearance of a curve for networks which have not been treated by our techniques is quite similar, showing the right hand size of a normal distribution curve in general, though the exact shape will depend on factors such as the number of naturally redundant units. The significance of our method lies in the ability to extend the curve further to the right, both in the very shallow slope region we have called robustness and in the steeper parts we have called damage resistance. If in the above diagram the area under the curve is representative of the usefulness of the network, then our method can increase that area considerably, under explicit user control.

Distinctiveness analysis can be used to add extra units which are coerced into duplicating the functionality of one or more existing units. For 100% redundancy, clearly a duplication of the entire network would suffice and would not require distinctiveness analysis. This method would not, however, avoid the problem of possible generalisation failures and the susceptibility to noise evinced by minimal networks. This method could not be extended to create a network with 50% robustness, by duplicating half of the existing units.

Doubling the size of the network by adding new units, each having a functionality being the compromise of a selected pair of original units would produce a network again with similar robustness, but with significantly increased noise resistance, and protection from generalisation failures.

Other levels of robustness can be added as required, largely determined by the number of units to be backed up, in that a pair of units which are more anti-parallel than orthogonal would produce a backup unit with less informational content than either original unit. In practice any required level of redundancy can be achieved, though sometimes requiring more than the optimum minimum number of extra units, for this reason. It is also possible to program in multiple redundancy in a hierarchical fashion to allow multiple failures to be protected.

A very important consequence of this directed post facto addition of backup units to provide robustness will be evident for use in specialised neural hardware as it becomes generally available. To whit, the backup units can be deliberately localised in a distant physical part of the hardware from the units being protected. This is of course important where real world phenomena are concerned, since damage will most likely affect a physical neighbourhood of units. We should therefore follow one of the most important maxims of backing up by keeping the backups well separated from the originals.

At this stage, the backup units are created with weight vectors composed from the weight vectors of the units being backed up. This is a better initial mapping into pattern space than random initialisations. A small step size random walk is performed with each backup unit to focus it on the appropriate location in pattern space. Any steps which do not reduce the distance to the destination are discarded. A more sophisticated search technique in pattern space was envisaged, however this turned out to be unnecessary. Even with this relatively inefficient method, the time taken for training the network is always noticeably longer than that required for this phase of the introduction of robustness.

Conclusion:

Our simple method for intoducing robustness rests on the base of being able to clearly distinguish differences in functionality of units. Since we can thus recognise the distinctiveness or otherwise of units in a feed-forward back-propagation network, we can explicitly control the speed and successfulness of learning by adding extra units when we automatically recognise that the network needs more units to learn; we can reduce the network to the minimum required to maintain functionality; and then introduce the desired level of redundancy.

The ability to design in specified levels of robustness and damage resistance will be very important for the use of neural networks in sensitive or life critical applications.

References:

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

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

Harris, D, Gedeon, TD, "On-line addition of units to nets which do not learn," Technical Report CSTR-91-7, Brunel University, 1991b.

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

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

Sietsma, J, Dow, RF, "Creating Artificial Neural Networks That Generalize," Neural Networks, vol. 4, pp. 67-79, 1991.