IMPROVING THREE LAYER NEURAL NET CONVERGENCE

Michael Alder, Sok Gek Lim, Paul Hadingham and Yianni Attikiouzel

The University of Western Australia, Australia

INTRODUCTION

The multi-layer Perceptron [1], the MADALINE [2], the committee net [3] and the feed forward networks trainable by the back-propagation algorithm, [4], are related in ways which are not always entirely clear. It is one aim of this paper to make some of them a little clearer.

The main application of neural networks has been as trainable pattern classifiers, and that is how they will be viewed in this paper. In the first part of this paper we investigate the relationship between three layer feed forward back-propagation nets (using the terminology of [4]) and the committee net of [3], and show that a simple modification to the algorithm of the latter makes them, in respect of their power to classify data sets, equivalent. Two algorithms may, however, be equivalent in power but differ greatly in their practicality. In the second part of the paper we conduct some experiments in order to determine whether the modified committee algorithm can compete with back-propagation in a variety of applications. It is found that the committee algorithm (a) is about 10 times as fast in some applications and (b) is much less prone to getting trapped in local minima.

The theoretical interest in the paper stems from the ease of analysing the committee algorithm together with the equivalence.The experimental interest is that this method of speeding up back-propagation may be used with other improvements to reduce training times in some applications.

In what follows we restrict attention, without loss of generality, to the case of a binary classification problem.

THEORETICAL CONSIDERATIONS

Terms not defined in this paper are to be found in the references, specifically see [3] for definitions of terms relating to committee nets. Briefly, a committee net is a (trainable) set of k oriented hyperplanes in the space each returning a 'vote' +1 or -1 according to which side of the hyperplane a datum is. The votes are summed by the 'output unit' and compared with a threshold of zero if the number of units is odd. By adding, if necessary, a unit or units remote from the data or by giving a threshold of 1/2, the case of an even number of units can be treated, as can thresholds other than zero.

The first proposition is the trivial observation, made for completeness:

Proposition. *Any binary data set can be correctly classified by a committee net.*

Proof: The argument is by induction on the number of points of the data set. We first observe that if there are only two points, one of each category, then the set is linearly separable and a single unit net will serve. Now suppose that there is always a correct classification of a data set having a total of r data points, and consider a dat set having r+1 points. One of the points at least must be extremal, i.e. may be separated from all the others by a hyperplane. The other points may be correctly classified by a committee. Without loss of generality, suppose the new point is positive and the committee which classifies the rest of the data set would assign it a negative value. Then we can insert between the old r points and the extremal point sufficiently many hyperplanes to make the new point positive. This will also add negative values to the r points which may alter the classification. We therefore take as many hyperplanes again which have all the points on the positive side. This returns the r points to the same values they had before, and preserves the positive sign of the extremal point.

□

Remark. There are better ways. In particular we need add only half the number of new hyperplanes suggested in the above argument. Even so, the argument is hardly cheering, suggesting as it does that the number of hyperplanes, and hence of hidden units, may increase with the size of the data set.

Remark. The 3-layer FFBP net may be regarded as a committee net with variable weights for the voting strengths together with a variable threshold. Again, the threshold is not essential as it can be replaced with a fixed threshold of zero and sufficiently remote units added in. Specifically, in $\mathbf{R}^n$ with input datum

$x = (x_1, x_2, \ldots x_n)^T$ the 3-layer FFBP net implements a map:

$$f(x) = \text{sgn}\left(\sum_j b_j \text{sgn}\left(\sum_i a_{ij} x_i\right)\right)$$

where the a_{ij} are the weights in the first hidden layer from the i^{th} component of the input to the j^{th} hidden unit and the b_j are the weights to the output unit, and the sgn function takes the value 1 if its argument is positive, -1 if it is negative and is not defined at zero.

Suppose that some data set is correctly classified by some such 3-layer FFBP net. We observe that since the data set is finite and $f^{-1}\{1\}$ is open, as is $f^{-1}\{-1\}$, there is stability under perturbations of the weights. Specifically we may take the b_j to be rational, which is just as well given the need for using these nets via computer simulations.

Moreover, if any of the coefficients should be negative, then reversing the sign of the corresponding a_{ij} weights will allow us to reverse its sign to still give a solution. And if a weight should be zero, we can simply remove the unit altogether. Hence we may take it that the weights are all positive rationals, and since multiplying throughout by any positive number will preserve the sign of the solution, we may multiply by the Least Common Multiple of the denominators to make the weights all positive *integers*, which we can then regard as multiplicities. This is now a committee net with some of the units replicated by their multiplicities.

The above argument leaves one with the impression that for any data set that can be classified by a 3-layer FFBP net there is a committee net which can classify it, but the number of units required for this equivalent committee can be arbitrarily large. Happily, such is not the case. We proceed to show this by a series of propositions motivated by some examples.

Example. Suppose we have a committee with multiplicities attached to the units: I shall write: (2, 11, 19, 26) to indicate that we have four units which we may take it correctly classify some data set in $\mathbf{R}^n$, provided we have the given multiplicities. Alternatively, we have 2+11+19+26 units but only four of them are distinct hyperplanes.

If n is big enough, then there will be some region of the space having a count of 2+11+19+26, and as we move across the last hyperplane we move into a region having a count of 2+11+19 -26, and so on. Now consider the second hyperplane. The regions on the positive side of it may have values ranging from 2+11+19+26 to -2+11-19-26, with similar values for the negative side. Now these numbers when added to +11 or -11 are from the set {-47, -43, -9, -5, 5, 9, 43, 47}. We observe that the sign of the result of adding 11 or subtracting 11 from these numbers will not be altered by changing the number 11 to some sufficiently close other number. In particular, choosing any number to replace 11 which is strictly between 9 and 43 will not alter any signs; this is, of course, that interval of the values obtained by taking sums and differences of the multiplicities which contains 11. We may therefore replace the given quartet (2,11,19,26) by the equivalent quartet (2,19,19,26) which must also correctly classify the same data set. Similarly, the 26 may be changed to any value between 2 and 36 so may also be made equal to 19. Finally the 2 may take any value between -19 and 19 in particular may take the value 0. We therefore have that the given quartet is equivalent to the quartet (0,19,19,19). This is of course equivalent to the quartet (0,1,1,1) or the triple (1,1,1). In other words, the original total of 58 units can be reduced to 3 and still classify the same data set.

Example. Suppose we have three units with multiplicities positive integers (a,b,c). We may, without loss of generality, suppose them arranged in non-decreasing order and not all the same. Now if c > a+b, the first two units cannot change the sign of a region by enough to compensate for the vote of the third unit and may be removed; the triple is equivalent to a single unit in the third location. Alternatively we may argue that b lies between a-c and c-a and can be put to 0, whereupon a lies between -c and c and can also be put to 0, finally divide (0,0,c) by c. If c< a+b then b lies between c-a and c+a and can be put equal to c, and finally a lies between 0 and 2c and hence may also be put to c, giving (c,c,c) ~ (1,1,1). Finally,the case where c = a+b is easily seen to be reducible to a triple (1,1,2) and requires a total of four units. Such a case can certainly arise from a 3-layer FFBP net, and we observe that in this case we need one extra unit in an equivalent committee.

Definition. Let $a = (a_1, a_2, \ldots a_k)$ be a set of non-negative integers. The *unitary span* of the set is the set of 2^k integers obtained by taking all linear combinations with coefficients in {-1,+1}. a is said to be *unitarily degenerate* iff 0 is in the unitary span. We write *uspan(a)* for the set of 2^k integers comprising the unitary span.

Definition. Let $a = (a_1, \ldots a_k)$ and $b = (b_1, \ldots b_k)$ be two k-tuples of non-negative integers in non-decreasing order. Let $w = (w_1, \ldots w_k)$ be a unitary vector i.e. each w_j is either -1 or +1. Then we write a~b iff for every unitary w, sgn(w*a) = sgn(w*b) where * denotes the usual inner product.

Remark. Clearly ~ is an equivalence relation and equally clearly if the k-tuples represent the *multiplicities of units in the hidden layer*, replacing one set of multiplicities with the other will not change the clasification of the net. If a k-tuple is degenerate then any equivalent k-tuple is also degenerate.

Definition. A k-tuple of non-negative integers in non-decreasing order will be referred to as a *multiplicity-tuple*.

Definition. If $a = (a_1,a_2,...,a_k)$ is a non-degenerate multiplicity-tuple, we write $\hat{a}_j$ for the sequence with a_j removed. Then if a' is the sequence a with a_j changed to b_j we say that the transformation from a to a' is *allowable* provided both a_j and b_j are in the same interval of consecutive elements of uspan($\hat{a}_j$), i.e. iff there is no element of uspan($\hat{a}_j$) between a_j and b_j.

Proposition. If a' is an allowable transform of a then a' and a are equivalent.

Proof: Suppose not. Then there is some unitary combination of the elements on which the sign of the result is different. Let the unitary vector involved be w, so sgn($w*a$) is different from sgn($w*a'$). Now the absolute value of the difference between $w*a$ and $w*a'$ is $|a_j - b_j|$. Suppose, without loss of generality, that $w_j = 1$ and let the residue r be given by $r = w*a - a_j = w*a' - b_j$ Since the sign of $r + a_j$ differs from the sign of $r + b_j$ there must be some number between a_j and b_j which is equal to r, which is an element of uspan($\hat{a}_j$). But this contradicts the definition of allowable transformations.

□

Remark. *Obviously if a' is an allowable transform of a, then a is an allowable transform of a'.* The computations of our first example demonstrate the ease with which one can reduce k-tuples by means of sequences of allowable transforms, and hence reduce the number of units required of a committee net while still preserving the classifying capacity. We now enquire, how many equivalence classes are there? This will allow us, with the above proposition, to find bounds on the number of units in a reduced committee equivalent to a given committee. When k = 3 for instance, we have seen that there are precisely two non-degenerate equivalence classes, which have representatives (0,0,1) and (1,1,1).

Definition. Any unitary vector w is *positive* if for every multiplicity tuple a, $w*a > 0$, it is *negative* if for every unitary vector a $w*a < 0$, and it is *ambiguous* if there are some a for which $w*a > 0$ and others for which $w*a < 0$.

Remark. It is clear that, say (1,1,.....1) is positive, and (-1,-1,...,-1) is negative, and that (-1,1,-1,1,...,-1,1) is positive. while (1,-1,1,-1,...,-1,1) is ambiguous. (-1,-1,1) is ambiguous, while (-1,1,1) is positive.

Proposition. For k odd, the number of equivalence classes of multiplicity- tuples is at least one greater than half the number of ambiguous unitary vectors of length k.

Proof: The equivalence classes are. by definition, the set of different consistent assignments to the 2^k different unitary vectors of the values +1 or -1. The values assigned to the positive and negative unitary vectors are forced to be +1 and -1 respectively, by definition. Consider first the case where the ambiguous vectors may be ordered by $u < v$ iff for every multiplicity-tuple a, $a*u < a*v$. Then we may take half the ambiguous vectors in this order and observe that an equivalence class is determined by choosing where to put a zero in the list. Since there are k/2 elements in the list there are 1+k/2 equivalence classes. If the vectors cannot be ordered, the usual state of affairs, there can only be more equivalence classes than this.

□

Remark. For small (odd) k, the comparisons are not unfavourable to the committee. For example when k = 3, the unitary vectors are (1,1,1),(-1,1,1) and (1,-1,1) for the positive vectors, three corresponding negative vectors, and the ambiguous vectors are (-1,-1,1) and (1,1,-1) which are paired. We may insert the zero before or after the (-1,-1,1) which gives two equivalence classes, (0,0,1) and (1,1,1).

For k = 5, ten of the unitary vectors are positive, ten negative and the remaining twelve vectors fall into six pairs. Among the six we find a chain of five and the last vector may occupy any of the first three places. There are thus 8 equivalence classes. It is easy to confirm that the following 8 multiplicity-tuples are inequivalent: (0,0,0,0,1), (0,0,1,1,1), (1,1,1,1,1), (0,1,1,1,2), (1,1,1,2,2), (0,1,1,2,3), (1,1,1,1,3) and (1,1,2,2,3). We observe that the maximum size of the equivalent committeee is therefore 9, and the average size is 5.5. In other words, by increasing the number of units in the hidden layer from 5 to 9 and locking the output weights to 1, we can guarantee to solve any problem solvable by a 3-layer FFBP net with 5 units (and others that cannot be solved!). It is found experimentally, as described in the next section, that the solution is obtained very much faster by the committee than by the full back-propagation net, and the system is far less likely to get trapped in local minima.

EXPERIMENTS

Remark. The training procedure for committees requires some analysis of the FFBP training procedure; an informal discussion of pertintent issues may be found in Fahlman [4]. Nilsson describes some procedures which have some defects. We briefly outline the crucial issues.

If the committee correctly classifies a point there is no need to do anything. If it is in error, we may take it that some of the units have to be corrected by the usual procedure for a single unit, the perceptron convergence procedure. The question is, which units is one to modify? Nilsson gives a heuristic argument for modifying the 'least wrong', that is to say, if $0 < a*x < b*x$ for the datum x and both a and b are in error and if only one unit needs to be adapted, then one adapts a. A rationale for this is that minimum changes are to be preferred since if the data set is mostly known, this is least likely to destroy 'knowledge about the data set already acquired by the net'. A drawback of this is that we may have an initial configuration in which all but one of the units are remote from all the data, and so only one unit is repeatedly adapted, the net is 'starved'. Strategies such as making random choices of which unit to correct, derived from annealing algorithm ideas, are alternatives.

We employ the tactic of correcting all units which are in error, but of adapting them by different amounts, the 'stepsize' being smaller for the more remote data. If we take a function such as $1/(1+x^2)$ for example, which decreases from 1 to zero, and use this as a stepsize for the case when the 'distance' x is $|a*y|$ (for y the augmented input vector), then this is precisely equivalent to weighting by the derivative of a sigmoid function arctan(x). This, of course, is precisely what is accomplished in the Back Propagation algorithm; the actual function arctan(x) appears essentially only via its derivative. We observe that the important thing about the derivative of sigmoid functions, is that their derivative for positive distances is always negative, the close neurons are moved more than the remote neurons, and hence the dynamic is not at any stage a contraction mapping. The repeated operation of different data on the system gives us, of course, an iterated function system.

The above observations suffice to specify a new committee net algorithm which we compared with standard back-propagation in a series of experiments conducted on a SUN SPARC1-station. First we treated the parity problem in dimensions 2,3,4 and 5, i.e. we took a unit hypercube in $\mathbf{R}^n$ and assigned alternate + and - categories to the vertices, for the cases of n = 2 to 5. In dimension 2 this is simply the XOR function. We took two 3-layer nets with various numbers of units in the hidden layer; first was the standard back-propagation algorithm and second was the case where the output units were held at weight one and only the hidden units corrected according to the above rule; we refer to this as 'simplified back-propagation' (SBP). Initial assignment of the weights was at random in the interval -3 to 3, and a number of repetitions of the training were run. The so called 'gain parameter' was varied but the dependency was not strong over a wide range of values. The averaged results are shown in table 1.

Table 1.

3-Layer Nets: The Parity Problem

Dimension	2	3	4	5
Net Geometry:	2-3-1	3-3-1	4-5-1	5-7-1
Moves				
B.P.	215	2842*	901*	3896*
S.B.P	46	195	760	1720
Time B.P.	0.03	0.9*	0.8*	13.0*
(secs) S.B.P.	0.002	0.016	0.2	1.3

* indicates that some runs failed to converge within 100,000 moves,

Different numbers of hidden units produced larger variations in numbers of moves and times for Back-Propagation than for Simplified B.P. as shown in table 2, where the data set was the parity problem in dimension 3:

Table 2

Net Geometry:	3-3-1	3-5-1	3-7-1
Moves B.P.	2842*	442	1632*
Moves S.B.P.	195	132	105
Time B.P. (secs)	0.9*	0.21	1.2*
Time S.B.P. (secs)	0.016	0.01	0.013

A more typical problem of an applied nature was used as a more realistic test. Some data obtained from an industrial collaborator relating to measurements

made on mineral samples was obtained. The vectors of length eight were obtained from an apparatus which reflected light from mineral samples and binned it into intervals of the spectrum. Colour gradings by eye were assigned to the two categories of data. Dividing the data into test data and training data has established that the accuracy of a back-propagation neural net is of the order of 78%. A three layer back propagation net was therefore trained until this level of error was reached. Similarly, a committee net was trained to the same level. The results for different numbers of units in the hidden layer are shown in table 3. Here we give run times on SUN SPARC stations required to obtain different per centages of the data correctly classified, 20 repetitions:

Table3.
Real Data: Three Layer nets.

BP Times, seconds:

	% of data set correct			
	75%	**77%**	**79%**	**81%**
Net Geometry				
8-1-1	427	947	*	*
8-3-1	253	1085	*	*
8-5-1	160	1037	*	*

Simplified BP Times, seconds:

	% of data set correct			
	75%	**77%**	**79%**	**81%**
Net Geometry				
8-1-1	23	65	94	*
8-3-1	36	43	88	77
8-5-1	-	-	-	748

- denotes no data collected, * denotes failure to converge.

CONCLUSIONS.

It might be argued that Back Propagation is so inefficient an algorithm that to attempt to improve it marginally is to waste time and energy. Improvements using, for example conjugate gradient and other types of net are known [5],[6]. We feel it is worth investigating the algorithm however for four reasons: first it constitutes something of a standard, if a low one; second some of the 'improvements' abandon all claim to being neural models, while some kind of a claim can still be made for the committee net. Third, as mentioned in the introduction, it is possible that some of the other improvements can be made in addition to our modification. In particular, nets with more layers can be handled using similar ideas, and in a subsequent paper we shall make similar modifications to four-layer nets. Finally, the committee net is much simpler to analyse; Back-Propagation conceals essential features of the dynamic.

For nets with a relatively small number of units in the hidden layer, improvements in time by a factor of about 20 are readily attainable. This is due to two factors: first the simplification of the algorithm entails less computation and second, the reduction in the size of the search space makes for fewer passes through the training set and reduces the risk of the net becoming trapped in a local minimum.

REFERENCES

1. Rosenblatt, F: 1968,. "The Perceptron" Spartan Books
2. Widrow B, Lehr M: 1990, 30 Years of Adaptive Neural Networks:Perceptron, Madaline,and Back-Propagation *Proc. IEEE September* 1415-1442
3. Nilsson, N. : 1965 "Learning Machines " McGraw-Hill
4. Rumelhart D, Hinton G and Williams R., August 1986 Learning Representation by Back-Propagation of Errors Nature, Vol 323 533
5. Fahlman S. and Lebiere C:1990, *The Cascade-Correlation Learning Architecture* in: D.S.Touretzky (Ed) "Advances in Neural Information Processing Systems 2 " Morgan Kauffman Publishers
6. Brent,R, 1990, " Fast Algorithms for Multi-Layer Neural Nets" The Australian National University Computer Sciences Laboratory Technical Report.